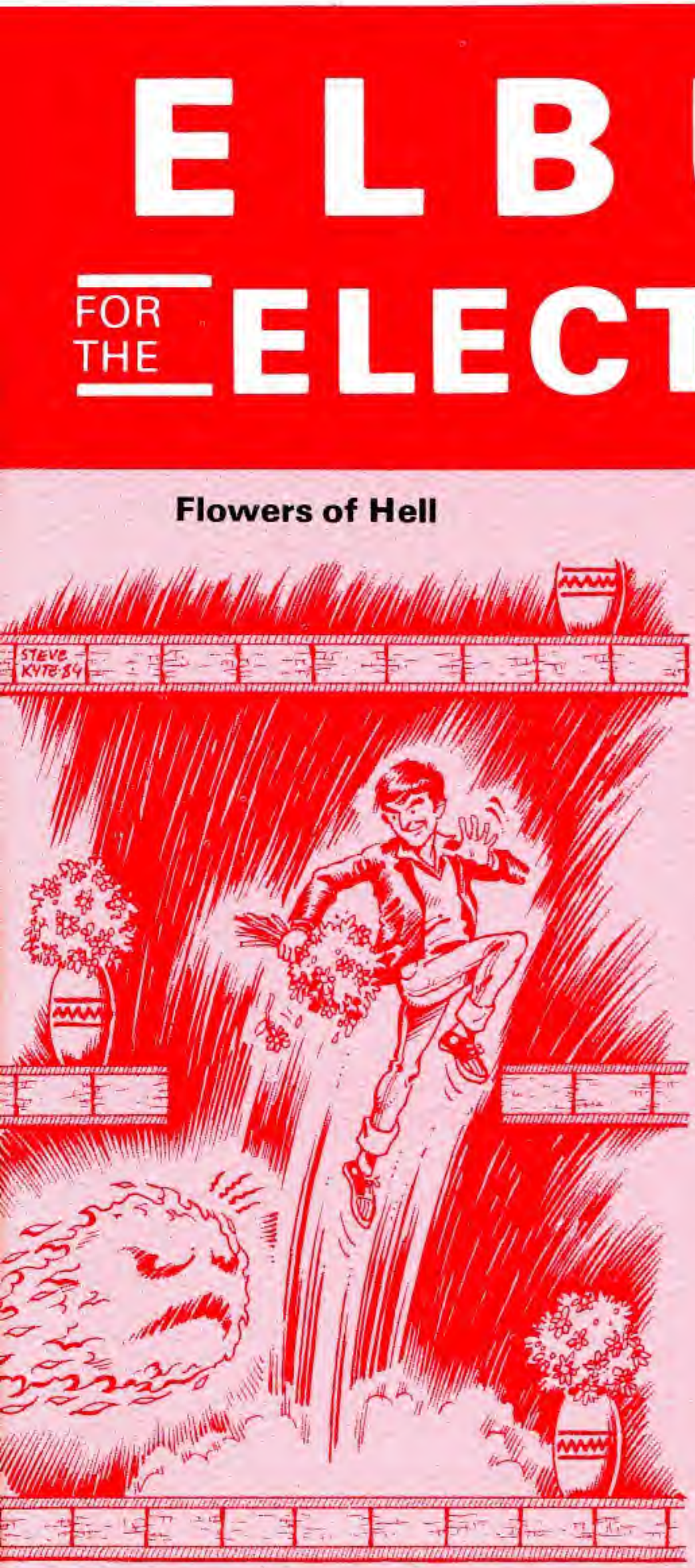
£1.00

ELBUG FOR THE ELECTRON

Vol 1 No 8 July 1984

Flowers of Hell



Games

* **Pontoon Card Game**
* **Electron Breakout**

Plus

* **Disco Lights**
* **Cassette Indexer**
* **Program Protection**
* **Debugging Programs**

Plus

* **Acorn's Plus 1 Reviewed**
* **Games Reviews**
* **Competition Results**
* **Hints and Tips**
* **And much more**

EDITORIAL

THIS MONTH'S MAGAZINE

The first add-on for the Electron from Acorn, called the Plus 1, is now available and we have managed to obtain one in time to include a review of this important development in this issue of ELBUG. As a consequence, we have had to postpone our review of TV/monitors to the next issue. This first add-on from Acorn is clearly an important event for Electron owners, and there may be more to come from Acorn in the future.

Cassettes are a cheap way of storing all your programs and other files, but keeping track of everything can be difficult. This is where the Tape Indexer in this issue will really come in handy. It will not only provide you with a list of all the programs on each of your cassettes, but it will also work out automatically the counter setting at the start of each program for faster access. We shall be covering some of the more general problems associated with cassette usage in the next issue of ELBUG magazine.

One of the subjects that occurs frequently in reader's letters is that of locating errors in programs. It is only too easy when typing in a long listing to introduce unnoticed errors which then subsequently prove difficult to find and correct. In the very first issue of ELBUG we published a short article describing the guidelines that we follow in preparing programs for publication, and we also gave a number of hints to help you along the way. Clearly the debugging of programs can be a difficult and time-consuming task and so we are publishing an article, in two parts, to provide you with further help and advice on this subject. This information can be just as useful, of course, for debugging your own programs as well as those typed in from magazine listings.

Games again feature strongly in this isuue with a version of the classic 'Breakout' game, an excellent screen display of 'Pontoon', and a very original game that we have entitled 'Flowers of Hell', featured on this month's front cover. In addition to these we have also included a short fun program for disco addicts under the heading of 'Electron Night Fever'.

CONTRIBUTIONS

We rely to some extent for the articles and programs that we publish in ELBUG on contributions sent in by our readers. If you have written any good games or other programs that you think would be of interest to other readers then we would be pleased to hear from you. A quick look at some of the articles and programs that we have previously published will show you the kind of format that is most suitable for the magazine.

Mike Williams

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOAR

Hint Winners

This month we have selected the hint by R.Lambley as the winner of the £10 prize, while the £5 prize goes to B.Miller-Smith. If you have any useful hints or tips that you think will be of interest to other readers then let us know. You might also win £10 for yourself.

MAGAZINE CASSETTE

Once again, all the programs from this month's magazine are also available on cassette to save time and sore fingers, and the details for ordering are given on the back cover. The cassettes for all previous issues of the magazine are also available at the same price. If you want to be sure of getting your magazine cassette at the same time as the magazine, then why not take out a subscription and save money as well.

ELBUG MAGAZINE

GENERAL CONTENTS

2 Editorial
4 Electron Expansion
The Plus 1 from Acorn
6 The Flowers of Hell
10 Give Your Electron Night Fever
12 Debugging Programs (Part 1)
14 More Games Reviewed
16 Cassette Indexer
20 Electron Breakout
23 Electron Graphics (Part 8)
26 Points Arising
27 Scramble Your Software
Program Protection
28 Screen Display Competition Results
29 Pontoon

HINTS, TIPS & INFO

9 Flexible Answers
9 Pressing Keys by Program
11 Back-up Copies of Machine Code Programs
11 Reserved Words
11 Changing the Bell Parameters
15 Using Tape Recorders with Automatic Record Level Control
15 Accurately Filling Rectangular Areas
19 Strange Crash
19 TAB Command
22 Shortened 'IF' Statement
22 Error Reporting and Editing
26 Clearing Screens with Windows
28 Phase Selection in Micro Power's "Croaker"
28 A Rounding Error?
33 The Hanging 'ELSE' Problem
33 Programming Caps Lock Mode

PROGRAMS

6 Flowers of Hell Game
10 Disco Lights
16 Cassette Indexer and Calibration Routine
20 Electron Breakout Game
23 Graphics Example
27 Program Protection Routine
29 Pontoon Game

ELECTRON EXPANSION — THE PLUS 1 FROM ACORN

by Nigel Harris

We have been eagerly anticipating the release of further add-ons from Acorn for their Electron as part of the overall design philosophy for this increasingly flexible machine. We have managed to obtain one in time to give you our first impressions.

Product : PLUS 1 expansion interface
Supplier : Acorn Computers Ltd.
Price : £59.90 inc VAT.

In previous issues of ELBUG (Vol.1, Nos.5,6) we have talked of add-ons for the Electron. So far, these have all come from outside Acorn. Now the originators of the machine themselves have moved into the sales arena, with the PLUS 1 expanding both the functions of their machine and therefore its possible areas of application.

Facilities that immediately become available simply by the connection of this new product are: a printer port (parallel), analogue port, and two ROM filing system cartridge sockets.

APPEARANCE

The PLUS 1 is a box with very similar cosmetic appearance to the Electron. It is the same width and height, and about three and a half inches from front to back. It's the same colour and texture of plastic, and when connected to the Electron matches it extremely well. Overall it appears to be very nicely made. Connection is made by plugging in to the expansion connector on the rear of the Electron, bringing the two cases into close contact. A couple of stout aluminium bolts render the union quite rigid, though care must be taken not to tighten these bolts too firmly since their purchase is against the plastic of the case.

Having fixed the two parts of the 'new' Electron together, it almost takes on the guise of a little BBC micro, but for the two slots on top which are to accept the ROM cartridges. Beneath sprung hatches, each of these six inch slots houses an edge connector not unlike the one that projects from the back of the computer. The ROM cartridge system gives the Electron the capacity for plug-in-and-run software. A complete program is permanently held in a ROM chip inside one of these small boxes. When the cartridge is pushed into the slot it becomes part of the system (until unplugged again).

USING THE INTERFACE

The machine must be switched off while anything is being plugged in, and this is true of the interface box as well. When the power is applied, nothing strange reveals itself immediately on the screen (no banners rejoicing in the extension of the basic machine). Typing *HELP, gives the interesting response:

Expansion 1.00
ADC/Printer/RS423

The analogue port (or analogue to digital converter - ADC) appears at the rear of the case as a 15 way D-type socket to accept the Acorn standard joystick plug. That means it is the same as the BBC micro's joystick interface but the PLUS 1's port does not make any provision for a light pen in the same way as the Beeb. So twin analogue joysticks may be plugged in here (they're the ones that are continuously variable between values instead of just on and off like the switched Atari type).

Printer connection is by the same standard parallel (Centronics type) 26 pin plug that the Beeb uses. The socket for this is mounted on the back of the new interface box for the Electron alongside the analogue port. It's a shame that the particular type of socket that was supplied on our review model did not have the eject levers that are normally used for unplugging a connector without exerting strain on the cable.

The last reference in the *HELP information is '/RS423'. RS423 is the standard by which some serial interfaces function. Yet there is not such a facility present in this box. Hinted at by Acorn but not available for a while at least, such an interface will make use of one of the cartridge ROM sockets in the PLUS 1 allowing serial printers or the use of modems (for communicating with other computers).

The initial intention for the use of the plug-in ROM cartridge system is for program packages and games. The first ones available include Acornsoft favourites like 'Tree of Knowledge', 'Lisp', and 'Snapper' and of course once plugged in, take no time at all to load - a vast improvement over tape systems and potentially much faster than a disc system. Cost of all these cartridges is £14.95 (inc. VAT) except for Lisp which costs £39.95.

An EPROM inside the PLUS 1 adds some extra *FX type commands to the operating system. These are to switch the printer on and off, the analogue port on and off, and for disabling the whole interface (so that the Electron thinks that the PLUS 1 is not there any more without having to physically disconnect them). Access to the values read at the analogue port inputs are available from within Basic using the ADVAL function and works in just the same way as with the Beeb (sorry to keep making these comparisons but it is inevitable).

With the new extension, an Electron is equipped to do many more jobs than it could before but the cost is a reduction in speed. With the interface connected you should expect a program to run about ten per cent slower than it would if not connected. This difference in most instances, will hardly be noticeable, but in case it is, Acorn have provided a new command - *FX163,128,0 - to disable the whole interface, as mentioned above. Enabling it again is done by a similar but inverse command, *FX163,128,1. It's rather a long and cryptic sort of thing to have to type but it is most effective.

EXPANDABILITY

Expandability is a popular concept and the first surprise when looking at the PLUS 1 is that there is no provision for extending the Electron's expansion interface any further. In other words once a PLUS 1 is connected to your machine, it will not support any other expansions like the ROM board from SIR Computers or the switched joystick interfaces that are beginning to appear from other manufacturers.

Calling this device the PLUS 1 leads one to speculate on the forthcoming existence of a PLUS 2 to be inserted between the Electron and the PLUS 1, thereby extending the expansion bus and making available some more useful facilities (like a disc filing system?). Perhaps one day there will be a whole family of PLUS's, each doing a different job.

CONCLUSIONS

This is a very attractive addition to Acorn's baby who now appears to be cutting its first teeth. How popular the device becomes remains to be seen; the combined facilities offered here are more than Acorn's competitors can currently supply but then if it were only joysticks that were needed then one could probably find a cheaper solution with one of these.

It seems strangely disproportionate these days; at one time the cost of a computer would have exceeded by far the price of a printer. Having paid something like £260 for an expanded Electron, you would be hard pushed to find many good printers comparably priced (the popular Epson printers are twice the cost of an Electron on average). I would personally have liked to see a disc interface emerge from the Acorn incubator but then (to over-use the metaphor) perhaps I am asking the baby to run before it can walk, and the popular demand is almost certainly for a package of this nature.

FLOWERS OF HELL

by C. E. H. Francis

You're trapped in the type of house that's probably an estate agent's nightmare. Your only escape is to get to the top floor from the level where you start. Below you is the cellar, and above are many levels, each of which must be quickly explored for a way through to the one above - unfortunately there are no stairs and you have to jump from floor to floor, but worse than this, the floors are unsafe and continually give way beneath you.

If you get to the top you are rewarded with a bonus round. Here you must repeat the same theme but score bonus points by saving the flowers from meeting a fiery fate when they tumble down through the house. If you don't stop them, then you lose points. There are two dangers. Firstly, do not get devoured by the fireballs that rush through the house on their way to the pit of Hades concealed beneath the cellar. Secondly, you must not fall through the floor of the cellar yourself. There is only one more place below that and it will engulf you - soul and all, even though you have three lives. Life is not a bed of roses; each bonus round successively reached will require you to catch more flowers to win points.

Controls from the keyboard are:

Z	-	left
X	-	right
,	-	jump up a floor
M	-	quantum jump (leap)
F	-	freeze game
U	-	unfreeze game

Jumping up a floor has to be done when a hole appears in the floor above, whereas a 'quantum jump' is defined as being a special jump that will go up to the next floor whether there is a hole or not. Obviously this is quite a powerful feature and so you only get one for the whole duration of the game.

PROGRAM NOTES

The program defines some special characters that you may like to see by typing PRINT CHR$ and a number from 224 to 230, and 240 and 241. PRINT CHR$226 is particularly interesting, and you can see how it is defined in the procedure PROCchar at line 1700. As a further point of interest (but only if you're the type who cheats) you might like to know that the variable C% holds the number of lives left. So by altering line number 150 you can set any number of lives you would like to start the game with, or alternatively, you could delete line 340, and you'll never lose any lives!

By keeping the variable names short in this program, the computer doesn't spend so much time interpreting the Basic code, so the program runs faster than it would, if the variable names were say half a dozen letters long. Unfortunately, this means that the program text is less readable than it might be.

→

```
   10 REM Program FLOWERS
   20 REM Version E1.2
   30 REM Author C.E.H.Francis
   40 REM ELBUG JULY 1984
   50 REM Program subject to Copyright
   60 :
  100 ON ERROR GOTO 490
  110 DIMGP%(32),FX%(2,10),FY%(2,10),Z%
(2,10),F$(2),FR$(2),FL$(2),FD$(2)
  120 PROCchar:PROCstrings
  130 H%=0
  140 REPEAT
  150 MODE1:S%=0:C%=3:PROCinstruct
  160 MODE5
  170 PROCsetup
  180 PROCsetup2
  190 REPEAT
  200 PRINTTAB(10,1);S%;SPC2;
  210 IF INKEY-103 AND GP%(Y%-1)=X% PRIN
TTAB(X%,Y%)MU$;:Y%=Y%-2:SOUND1,-15,96,1
  220 IF INKEY-102 AND QJ% THEN QJ%=FAL
SE:PRINTTAB(X%,Y%)MU$;:Y%=Y%-2
  230 IF INKEY-67+INKEY-98=0 PRINTTAB(X
%,Y%)M$;
  240 IF INKEY-67 AND X%<19 PRINTTAB(X%
,Y%)MR$;:X%=X%+1
  250 IF INKEY(-98) AND X%>0 PRINTTAB(X
%,Y%)ML$;:X%=X%-1
  260 IF GP%(Y%+1)=X%:PROCdown
  270 Q%=2+2*RND(14)
  280 PRINTTAB(GP%(Q%),Q%)F$;
  290 PROCgap(Q%)
  300 IF A% PROCF
  310 IF INKEY-68 REPEAT UNTIL INKEY-54
=-1
  320 UNTIL Y%=3 OR Y%=31 OR T%=1
  330 IF Y%=3 PROCbonus:GOTO180
  340 C%=C%-1
  350 IF Y%=31 PROCbottom
  360 IF T%=1 PROCfire
  370 IF C% THEN 160
  380 MODE1
  390 PRINTTAB(10,3)"GAME OVER"
  400 IF H%<S% THEN H%=S%
  410 PRINT''"YOU SCORED ";S%
  420 PRINT''"HI-SCORE   ";H%
  430 PRINT''''"Would you like another g
ame? (Y/N)";
  440 REPEAT:G$=GET$:UNTIL INSTR("YyNn"
,G$)
  450 UNTIL G$<>"Y" AND G$<>"y"
  460 MODE6
  470 END
  480 :
  490 ON ERROR OFF:MODE6
  500 IF ERR<>17 REPORT:PRINT at line "
;ERL
  510 END
  520 :
 1000 DEF PROCdown
```


```
 1010 SOUND1,-15,0,1
 1020 PRINTTAB(X%,Y%)MD$;
 1030 Y%=Y%+2
 1040 PROCgap(Y%-1)
 1050 ENDPROC
 1060 :
 1070 DEF PROCbonus
 1080 VDU19,3,14,0,0,0
 1090 VDU19,0,13,0,0,0
 1100 PRINTTAB(15,1);"BONUS"
 1110 VDU19,2,11,0,0,0
 1120 RESTORE1140:REPEAT
 1130 READI%,J%:SOUND1,-15,I%,J%
 1140 DATA 56,5,0,1,56,5,0,1,60,2,0,1,6
0,2,0,1,60,2,0,1,60,2,0,1,68,5,0,1,68,5
,0,1,76,12
 1150 UNTIL I%=76
 1160 IFA%>0THEN:FORI%=0TO2:FORJ%=1TOA%
:PRINTTAB(FX%(I%,J%),FY%(I%,J%))SPC1:NE
XT:NEXT
 1170 PROCdelay(200)
 1180 A%=A%+1:S%=S%+50*A%
 1190 IFA%>10THENA%=10
 1200 PRINTTAB(X%,Y%)SPC1
 1210 VDU19,2,4,0,0,0:VDU19,3,5,0,0,0
 1220 PRINTTAB(15,1);SPC5;
 1230 VDU19,0,3,0,0,0
 1240 ENDPROC
 1250 :
 1260 DEF PROCbottom
 1270 SOUND1,1,92,60
 1280 CLS:COLOUR1
 1290 PRINTTAB(2,5)"YOU FELL THROUGH "
 1300 PRINTTAB(2,8)"THE BOTTOM FLOOR"
 1310 D%=200:PROClives
 1320 A%=0
 1330 ENDPROC
 1340 :
 1350 DEF PROCF
 1360 FORI%=0TO2:FORJ%=1TOA%
 1370 IFX%=FX%(I%,J%)ANDFY%(I%,J%)=Y%PR
OCcaught
```

```
 1380 IFZ%(I%,J%)<0PRINTTAB(FX%(I%,J%),
FY%(I%,J%))FL$(I%);ELSEPRINTTAB(FX%(I%,
J%),FY%(I%,J%))FR$(I%);
 1390 FX%(I%,J%)=FX%(I%,J%)+Z%(I%,J%)
 1400 IFFX%(I%,J%)=0ORFX%(I%,J%)=190RRN
D(20)=1THENZ%(I%,J%)=-Z%(I%,J%)
 1410 IFFX%(I%,J%)=GP%(FY%(I%,J%)+1)PRI
NTTAB(FX%(I%,J%),FY%(I%,J%))FD$(I%);:FY
%(I%,J%)=FY%(I%,J%)+2
 1420 IFX%=FX%(I%,J%)ANDFY%(I%,J%)=Y%TH
ENPROCcaught
 1430 IFFY%(I%,J%)=31THENPROCFbottom
 1440 NEXT:NEXT
 1450 ENDPROC
 1460 :
 1470 DEF PROCcaught
 1480 IFI%SOUND1,-15,68,4:SOUND1,-15,80
,4:SOUND1,-15,96,4:SOUND1,-15,80,4:S%=S
%+5*I%:FY%(I%,J%)=3:PRINTTAB(X%,Y%)CHR$
226:ELSET%=1
 1490 ENDPROC
 1500 :
 1510 DEF PROCsetF
 1520 FORI%=0TO2:FORJ%=1TOA%:FY%(I%,J%)
=3:FX%(I%,J%)=RND(18):Z%(I%,J%)=2*RND(2
)-3
 1530 NEXT:NEXT
 1540 ENDPROC
 1550 :
 1560 DEF PROCFbottom
 1570 PRINTTAB(FX%(I%,J%),FY%(I%,J%))SP
$;:FY%(I%,J%)=3
 1580 IFI%THENS%=S%-5*I%:SOUND1,-15,32,
4:SOUND1,-15,48,8:SOUND1,-15,32,4
 1590 ENDPROC
 1600 :
 1610 DEF PROCfire
 1620 SOUND0,-15,6,15
 1630 CLS:COLOUR1
 1640 PRINTTAB(2,5)"YOU GOT CAUGHT"
 1650 PRINTTAB(2,8)"BY A FIREBALL"
 1660 D%=300:PROClives
 1670 T%=0:A%=0
 1680 ENDPROC
 1690 :
 1700 DEF PROCchar
 1710 ENVELOPE1,50,-8,0,0,5,0,0,126,0,0
,-126,126,126
 1720 ENVELOPE2,157,1,0,0,8,0,0,126,0,0
,-126,126,126
 1730 VDU23,224,255,255,255,254,252,192
,192,128
 1740 VDU23,225,255,66,66,0,0,0,0,0
 1750 VDU23,226,24,24,60,90,24,60,36,102
 1760 VDU23,227,255,255,255,127,63,3,3,1
126,24
 1780 VDU23,229,0,16,84,56,254,56,84,16
 1790 VDU23,230,0,8,42,28,127,28,42,8
 1800 VDU23,240,12,12,60,47,12,14,58,35
 1810 VDU23,241,48,48,60,244,48,112,92,
196
 1820 VDU23,1,0;0;0;0;
 1830 ENDPROC
 1840 :
 1850 DEF PROCinstruct
 1860 VDU19,3,3,0,0,0:VDU19,2,6,0,0,0
 1870 VDU28,0,2,39,0:COLOUR130:CLS:VDU26
 1880 COLOUR1:PRINTTAB(10,1)"THE FLOWER
S OF HELL":COLOUR128
 1890 COLOUR2:PRINT''"Move the man ";
 1900 COLOUR3:VDU226
 1910 COLOUR2:PRINT" up through the gap
s in "'"the floors. But he falls throug
h the"'"gaps too. Catch flowers ";
 1920 PRINTF$(2)+CHR$32+F$(1);
 1930 PRINT" to score."
 1940 PRINT'SPC11CHR$229" - SCORES 5"
 1950 COLOUR3:PRINT SPC11 CHR$229" - SC
ORES 10"
 1960 COLOUR2:PRINT''"If flowers drop th
rough bottom floor"
 1970 PRINT'SPC11 CHR$229" - LOSES 5"
 1980 COLOUR3:PRINT SPC11 CHR$229" - LO
SES 10"
 1990 COLOUR2:PRINT'"Avoid fireballs ";
 2000 COLOUR1:PRINT CHR$228;:COLOUR2:PR
INT" and do not fall"'"through the bott
om floor. Reach the"'"top floor for a b
onus."
 2010 PRINT"To move man use keys:"
 2020 PRINT'SPC7"Z - MOVE LEFT"
 2030 PRINTSPC7"X - MOVE RIGHT"
 2040 PRINTSPC7", - JUMP UP"
 2050 PRINTSPC7"M - QUANTUM JUMP"
 2060 PRINTSPC5"F/U - FREEZE/UNFREEZE"
 2070 PRINT''"Use keys together to avoid
 falling back"'"after jump. Only one qu
antum jump"'"allowed per man, ";C%;" me
n per game."
 2080 COLOUR3:PRINT'SPC6"PRESS ANY KEY
TO CONTINUE";
 2090 G%=GET
 2100 ENDPROC
 2110 :
 2120 DEF PROCsetup
 2130 VDU23,1,0;0;0;0;
 2140 VDU19,0,4,0,0,0
 2150 VDU19,2,2,0,0,0:VDU19,3,5,0,0,0
 2160 A%=0:QJ%=TRUE:COLOUR2
 2170 FORQ%=4TO30STEP2
 2180 PRINTTAB(0,Q%)CHR$224+STRING$(18,
CHR$225)+CHR$227;:PROCgap(Q%)
 2190 NEXT
 2200 ENDPROC
 2210 :
 2220 DEF PROCsetup2
 2230 X%=0:Y%=27
 2240 IF A%>0 THEN PROCsetF
 2250 COLOUR1:PRINT TAB(4,1)"SCORE"
```

```
 2260 IFC%PRINTTAB(0,1)STRING$(C%-1,M$);
 2270 PRINT TAB(X%,Y%) M$
 2280 ENDPROC
 2290 :
 2300 DEF PROCstrings
 2310 C$=CHR$17:R$=C$+CHR$1:B$=C$+CHR$2
:Y$=C$+CHR$3
 2320 N$=CHR$11:W$=CHR$8:E$=CHR$9:S$=CH
R$10
 2330 NNW$=N$+N$+W$:SW$=S$+W$
 2340 F$=B$+CHR$225:M$=Y$+CHR$226
 2350 SP$=CHR$32:MU$=SP$+NNW$+M$
 2360 MD$=SP$+SW$+F$+SW$+M$
 2370 ML$=W$+Y$+CHR$241+SP$
 2380 MR$=SP$+Y$+CHR$240
 2390 F$(0)=R$+CHR$228
 2400 F$(1)=B$+CHR$229
 2410 F$(2)=Y$+CHR$230
 2420 FORI%=0 TO 2
 2430 FL$(I%)=W$+F$(I%)+SP$
 2440 FR$(I%)=SP$+F$(I%)
 2450 FD$(I%)=SP$+SW$+S$+F$(I%)
 2460 NEXT
 2470 ENDPROC
 2480 :
 2490 DEF PROCgap(I%)
 2500 GP%(I%)=RND(18)
 2510 PRINTTAB(GP%(I%),I%)SP$;
 2520 ENDPROC
 2530 :
 2540 DEF PROCdelay(T%)
 2550 TIME=0:REPEATUNTILTIME>=T%
 2560 ENDPROC
 2570 :
 2580 DEF PROClives
 2590 IF C%=1 THEN PRINT TAB(1,11)"ONLY
ONE LIFE LEFT"
 2600 IF C%=2 THEN PRINT TAB(1,11)"ONLY
TWO LIVES LEFT"
 2610 IF C%=0 THEN PRINT TAB(3,11)"NO L
IVES LEFT"TAB(5,14)"GAME OVER"
 2620 PROCdelay(D%)
 2630 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

FLEXIBLE ANSWERS - B.Miller-Smith

A way to cover most eventualities when requesting a yes or no answer is illustrated in the following program

```
100 yes$="Y y YES Yes YEP YUP OK I WOULD I would":REM extend as required
110 no$="N n NO No NOPE NOT NOW NO THANK YOU":REM Ditto
120 test$=yes$+no$
130 INPUT "Do you want instructions ?"answer$
140 reply%=INSTR(test$,answer$)
150 IF reply%=0 THEN PRINT "I'm sorry, I didn't understand that":GOTO 130
160 IF reply%<LEN(yes$) THEN reply%=1 ELSE reply%=0
170 REM Exits with reply%=1 for "YES", 0 for "NO"
```

The string yes$ should contain all the most likely occurrences of words meaning 'yes' and the variable no$ the same for the 'no' answers. A response input as answer$ at line 130 is then tested at line 140 such that the value of reply% indicates whether a 'yes' or 'no' answer was received.

PRESSING KEYS BY PROGRAM - R.Sterry

You can use the *FX138,0,n command to enter a character in to the computer's keyboard buffer where n is the ASCII code for the character concerned; the computer then behaves as if the corresponding key had been pressed. For example, the ASCII code for the letter L is 76, so *FX138,0,76 inserts L into the keyboard buffer. You could try putting the following few lines into a program.

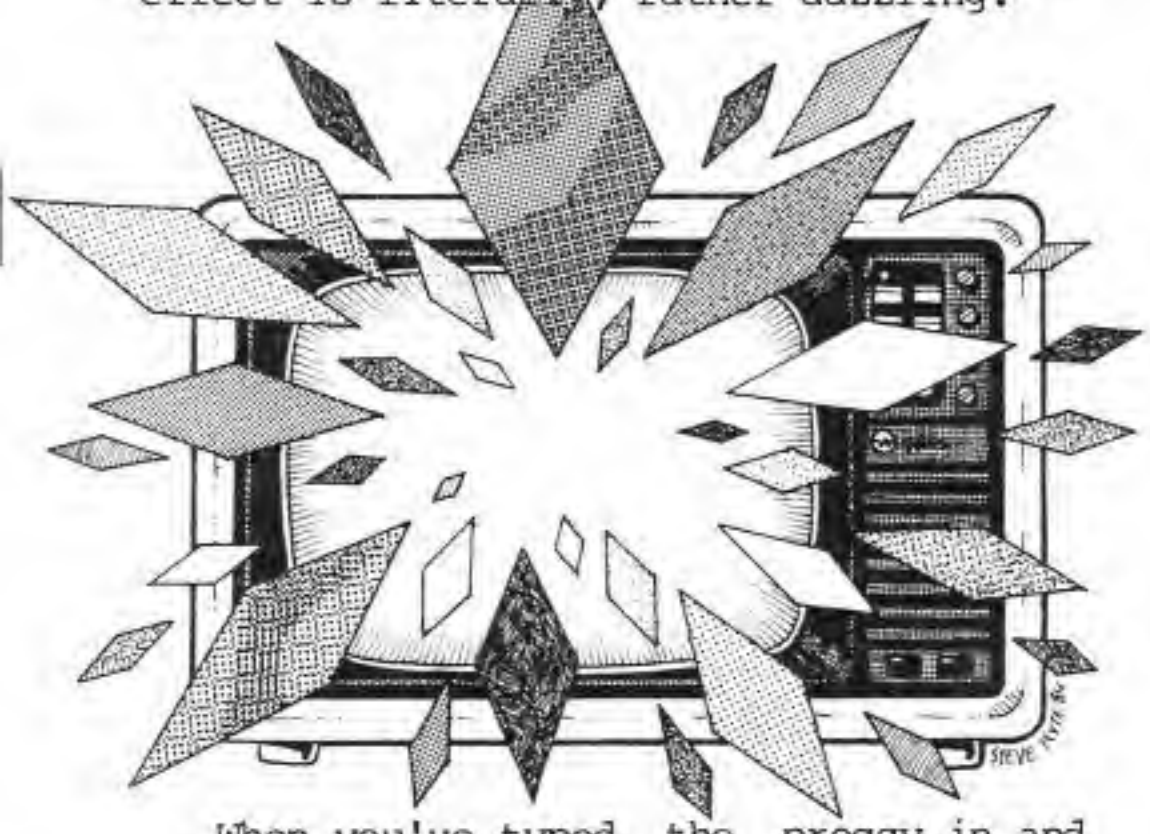
```
10000 *FX15,0
10010 *FX138,0,76
10020 *FX138,0,73
10030 *FX138,0,83
10040 *FX138,0,84
10050 *FX138,0,13
10060 END
```

This inserts the characters for the command LIST followed by Return into the keyboard buffer, and when the program terminates, Basic takes its input from there. The contents of the keyboard buffer are only used when the computer expects characters from the keyboard, and it responds just as though someone had typed them in. (*FX15,0 is used to clear the buffer initially of any superfluous characters).

GIVE YOUR ELECTRON NIGHT FEVER

by M. Hayes & D. McLoughlin

Here's a fascinating program for your Electron. We're not quite sure how it works but it obviously does and the effect is literally, rather dazzling.

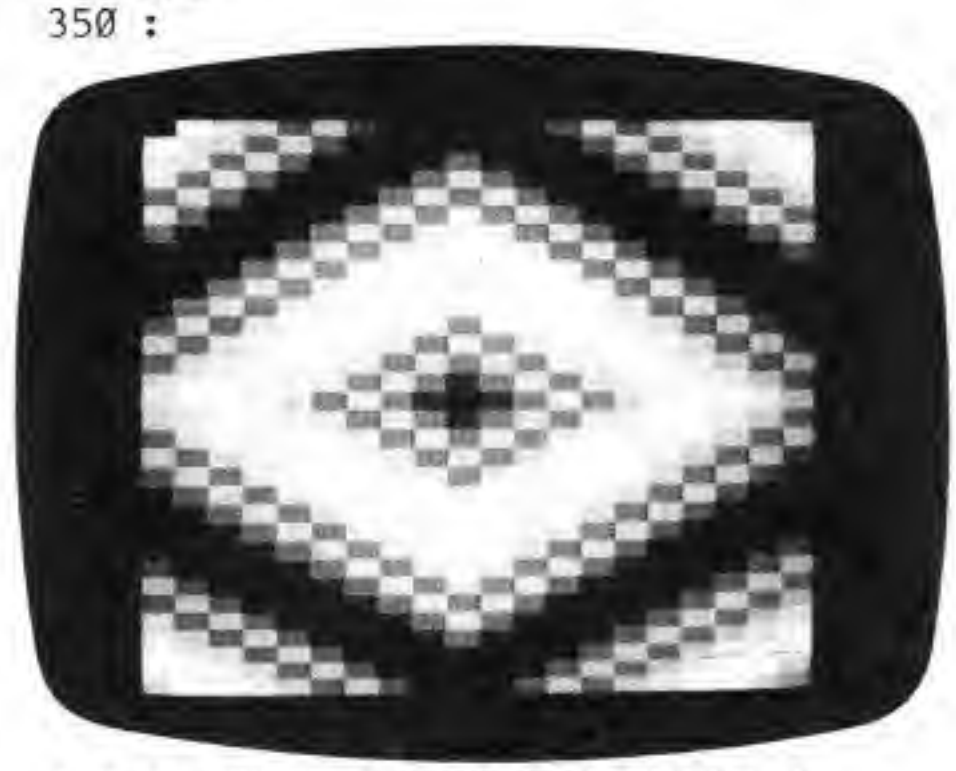


When you've typed the proggy in and saved it on cassette, put on a tape of your raunchiest music in the Electron's cassette player and run the program. You'll find that the display will jump into life as soon as the music starts. You need to be able to hear the music that's playing to the computer for the best results - some cassette recorders switch their sound off when they are plugged in. The display will change to the rhythm of the tune that you put on. This is well worth seeing in colour, so you'll have to make some other arrangements if you usually work in black and white.

The way that the program works is simply by sampling the signal that arrives at the cassette port and adjusting the video display according to a set of rules provided by the program. The main part of it is quite short. This is so that the program will run as quickly as possible for the computer to respond effectively to the music coming in from the cassette.

Now your Electron is all set for a spell of Saturday night fever.

```
 10 REM Program DISCO LIGHTS
 20 REM Version E1.3
 30 REM Authors M.Hayes & D.McLoughlin
 40 REM ELBUG JULY 1984
 50 REM Program subject to Copyright
 60 :
100 MODE1
110 ON ERROR GOTO 360
120 PROCtitle
130 MODE2
140 FOR C%=0 TO 15
150 VDU19,C%,0,0,0,0
160 NEXT
170 FOR L%=0 TO 638
180 COLOUR 128+ABS(10-L% MOD 20)+ABS(
16-L% DIV 20)
190 VDU32
200 NEXT
210 VDU23;11;0;0;0;0;
220 *MOTOR 1
230 REPEAT
240 VDU19,L%,0,0,0,0
250 P%=0
260 REPEAT
270 P%=P%+1:M%=?&FE04+?&FE09
280 UNTIL (P%=100) OR (M%<>N%)
290 IF P%=100 THEN FOR P%=0 TO 15:VDU
19,P%,0,0,0,0,:NEXT:GOTO250
300 N%=?&FE04+?&FE09
310 VDU19,L%+N% DIV 64,N% MOD 8,0,0,0
320 L%=(L%+N%)MOD15
330 UNTIL FALSE
340 END
350 :
```



```
360 ON ERROR OFF:MODE 6:*MOTOR 0
370 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
380 END
390 :
1000 DEF PROCtitle
1010 VDU28,0,2,39,0:COLOUR130:CLS
1020 COLOUR1:PRINTTAB(13,1)"Disco Ligh
ts"
```

```
 1030 VDU26:COLOUR128
 1040 PRINTTAB(17,4)"by":PRINTTAB(12,6)
"Matthew Hayes":PRINTTAB(16,8)"and":PRI
NTTAB(10,10)"David Mcloughlin"
 1050 COLOUR2
 1060 PRINT''''"This program displays a
regulated set of":PRINT"patterns,  acco
rding  to the beat of the"
 1070 PRINT"music.  This music must be
inserted into":PRINT"your cassete recor
der, and played at the"
 1080 PRINT"volume for loading programs
."
 1090 COLOUR1
 1100 PRINT''"THIS MUSIC MUST BE NORMAL
 MUSIC, AND NOT":PRINT"COMPUTER GENERAT
ED MUSIC."
 1110 COLOUR3:VDU19,3,2,0,0,0
 1120 PRINTTAB(6,30)"Press any key to c
ontinue";
 1130 G=GET
 1140 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

BACK-UP COPIES OF MACHINE CODE PROGRAMS - D.Bonehill

To make a back-up copy of your favourite cassette (not protected), type *OPT1,2 <return> followed by *LOAD"" <return>. Press 'PLAY' on your cassette recorder to load the program to be backed up. When this is complete, the screen will display a series of numbers, for example: GENOCIDE 08 0860 00003400 00000E00

Load a new cassette and then type: *SAVE GENOCIDE E00 + 860 3400 where the first and second numbers are the second and third numbers from above but simply reversed in order. Press 'PLAY' and 'RECORD' and you will be able to make a back-up copy of many machine code as well as Basic programs.

Remember that if a program consists of several parts, then each part will need to be backed up separately. Note that this method cannot be used to make back-up copies of protected software.

RESERVED WORDS - R.R.Hull

Take care when using capital letters for variable names that the names don't coincide with any that Basic uses like COUNT or PRINT because you will otherwise get errors with your programs. The best solution is to use lower case letters for variable names which will avoid any such problems.

CHANGING THE BELL PARAMETERS - K.B.Kealy

Most computers have a simple sound generator of some sort. Originally it used to be a bell, for compatibility with the old electronic keyboard standards, hence the name used now. Nowadays, in common with most computers, the Electron contains a more sophisticated electronic version. If you hold down the control key and press 'G', you will hear the Electron's 'BELL'. Similarly, VDU 7 in a program will have the same effect. However, there are four FX calls which allow you to alter the sound of the BELL in a similar manner to the SOUND command:

*FX 211,n - Selects the sound channel used for the BELL. Default is 1.

*FX 212,n - Sets the volume of the BELL if a negative value of n is used, or an envelope if n is positive. If an ENVELOPE is specified, then the value of n should be set to (ENVELOPE no.-1)*8. Similarly, for the amplitude, subtract 1 and multiply by 8. Default setting is 144.

*FX 213,n - This sets the frequency of the BELL. Default is 101

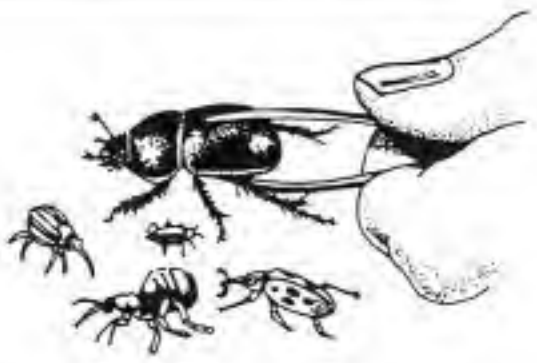
*FX 214,n - Finally, the duration of the BELL is set with this FX call. Default setting is 7. For example try running the following program:

```
10 ENVELOPE 1,2,3,-2,5,7,5,12,63,-30,0,-39,126,63
20 *FX211,1
30 *FX212,0
40 *FX213,100
50 *FX214,10
60 VDU 7
```

Now, whenever an error occurs, or the ASCII code 7 is intercepted, the BELL will be sounded with the parameters set up in the program.

DEBUGGING PROGRAMS (Part 1)

by Mike Williams



In this the first of two articles on debugging programs, we look at some of the problems that may arise from typing in programs from printed listings.

One of the most common subjects in letters received from ELBUG members, is that of programs listed in the magazine. Generally the contents of such a letter will be along the lines of:

"...having spent hours typing in the Block Blitz program from the March issue, I keep getting the message "No such variable at line...". I have checked this line several times with the printed version and I can find nothing wrong. Can you tell me if there is a bug in the program or am I missing something?...".

I have now answered many such queries, and have a pretty good idea of what goes wrong. Although there were just a few errors in the listings of some programs in early issues of ELBUG, these were not serious and were notified in the following issues. The majority of these letters clearly relate to errors that have been introduced during the process of typing the program into the Electron from the printed listing.

MAGAZINE STANDARDS

In the first issue of ELBUG I wrote about the standards that we adopt in printing program listings in the magazine. For the benefit of more recent members, it is probably worth repeating the more important points here.

As we said then, all the listings are printed directly from working versions by first transferring the program concerned from the Electron to a BBC micro. We try to make sure that the programs we print are both good examples of the programmer's art, and also clearly presented to minimise errors when the program is being typed in. You are recommended to keep to the same line numbers as in the original listing, and to incorporate all apparent spaces except the one immediately after the line number. Doing this will help to reduce the possibility of error.

Spaces can be particularly troublesome, as being non-printing characters, it is sometimes difficult to determine exactly how many spaces are needed. Use adjacent characters if necessary to help count out the number of spaces involved and remember that all programs are normally printed with WIDTH set to 39 (to give the same column width as for text).

We also try to ensure that variables, functions and procedures are all given meaningful names using lower case letters. It is not always possible to achieve this - other constraints, such as the use of the integer variables A% to Z%, sometimes dictate otherwise.

FINDING THE BUGS

The need for accuracy when typing in a program cannot be stressed too strongly. Even so, typing in long programs is tiring and it is only too easy to introduce the odd unnoticed error as you go along. One of the commonest errors, as already mentioned, is that of "No such variable at line...". Sometimes, on checking, you will find that a typing error has resulted in an incorrect variable name. Correct this and the error goes away. Very often, however, careful checking of the line referred to reveals no apparent error.

There are now two possible explanations. When dealing with variables, it is important to understand that they can be used in two somewhat different ways. A variable can be used in an arithmetic expression for example, or in a PRINT statement or similarly in a variety of other instructions. In each case it is the value, either a number or a character

string, currently assigned to the variable that is used in this way. If you try to use a variable to which the program has not yet assigned some value, then the message "No such variable..." will result. When this message occurs, it usually means that there is an error earlier in the program, probably in the line that should have assigned a value to this variable.

If the line that generates the error message contains more than one variable it is a simple, if slightly tedious matter, to determine which variable is actually responsible for the error message. As soon as the error occurs, list out that line, and then (in immediate mode) just type PRINT followed by the name of one of the variables and Return. Repeat this for each of the variables listed in turn. The first one to cause a repeat of the error message is the one that is causing the trouble.

Once you have got this far, you must then check the program, and try and find where this variable is assigned a value. Because of the way programs are structured, this will quite often be in a procedure called PROCinit, or PROCstart or something similar. It is these lines that then need careful checking with the printed version of the program, rather than the line that was first flagged as being in error.

The second possible explanation for this situation is rather more subtle, and requires some understanding of how Basic distinguishes between variable names and keywords in a program. A Basic keyword is one of the Basic instructions GET, THEN, FOR, etc. The Electron's version of Basic, the same as used by the BBC micro, allows variable names to be of any length and to include embedded keywords, although they cannot start with a keyword. Because they may be embedded within variable names, it is possible for Basic to allow a variable name "NAMELIST" or "GIFT", where "LIST" and "IF" are not read as commands, but simply as part of the variable name. This is not true of most other micros, which would recognise keywords wherever they occurred.

Now BBC Basic is much more flexible in this respect, so that you don't have to keep checking, when choosing variable names, to see if part of the name is a forbidden keyword. In addition, any confusion can be completely avoided by using only lower case for variable names, which we try to do in our programs, as keywords are only ever recognised when in upper case.

However, there are still some situations where confusion can occur, and this involves particularly the keyword "THEN". The following lines will help to illustrate:

```
10 IF A=BTHEN GOTO 100
10 IF A=B THEN GOTO 100
10 IF A=B%THEN GOTO 100
```

On the Electron, Basic will interpret the first of these lines as having a variable called "BTHEN", where the apparent keyword "THEN" is embedded as part of the variable name "BTHEN". The second line is interpreted as expected, with the space after the variable "B" indicating that this is the end of the variable name. However, if an integer variable is used, the percentage sign will indicate the end of the name, and no further space is required.

Although the use of lower case variable names is to be preferred, this can also have the effect of obscuring errors of this kind. Consider the following:

```
10 IF abcd=abcdTHEN PRINT "Yes"
```

This is still in error, since it is allowable to mix upper and lower case characters in variable names, and thus the line above specifies a variable "abcdTHEN" following the '=' sign. It is a good idea to ensure that keywords like THEN and ELSE are always preceded by a space to avoid the possibility of error.

The moral of this is that spaces can significantly change the meaning of an instruction if used incorrectly (either left out when needed, or added when not needed). This is again a reason for recommending that you try to type in any printed program as accurately as possible, including spaces. Next month we will look at some more tips for debugging programs, including some for your own programs.

MORE GAMES REVIEWED

Title : Intergalactic Trader
Supplier : Program Power
Price : £8.95 inc VAT
Reviewer : Nigel Harris
Rating : **

This is not an arcade game. It describes itself as a text only game for up to nine players and may well take an evening to play. It's really a kind of elementary Monopoly in space. Of course it didn't actually need to be in space but I think the authors supposed that this adds extra appeal for the kind of audience that they are trying to cater for.

A target figure is agreed with the other competitors before the game starts and you then race to achieve this target. To do this you must mine the resources of an asteroid that you're given at the start of the game. The transportation necessary to enable the sale of these resources costs a considerable amount. One asteroid does not yield enough and you would be expected to have to buy several in order to compete. Along the way there are many hazards that appear in a random fashion much like the chance cards in Monopoly.

The game is written in Basic, and for once is 'listable' for those interested in seeing how a large program like this might be written. It's important to notice that this is not an action game and may seem a little dull to most. However, it seems to be quite well thought out and may appease the Monopoly maniacs who are waiting to see their favourite game on the Electron.

Name : Electron Invaders
Supplier : Micro Power
Price : £7.95 inc VAT
Reviewer : David Fell
Rating : ****

Last month I reviewed Invaders from Superior Software, and said that I couldn't fully recommend it as a game. This month I've been playing Electron Invaders from Micro Power (formerly Program Power). I found this game far more enjoyable than Superior's, as it featured a number of variations on the basic theme of invaders and was well planned and designed.

There are options within the game to freeze, restart, turn the sound on and off, and to play a one or two player game. As the game progresses, different types of alien advance down the screen, with more vicious and higher scoring invaders appearing at the top. There are three sorts of missile - slow and fast normal bombs, and a type of bomb that splits into two pieces of shrapnel that fly off at 45 degrees. This sort can be particularly effective, and require particular respect!

Overall the game is a superb version of the classic arcade game, and I recommend its purchase. About my only criticism is that the graphics occasionally flicker - but the game doesn't appear to slow down, no matter what's happening on the screen.

→

Title : Percy Penguin
Supplier : Superior Software
Price : £7.95 inc VAT
Reviewer : Nigel Harris
Rating : ***

Percy is a close relative of that other antipodean of the arcade games, Pengo. He is trapped in a maze of ice blocks with the dreaded Snobees who will kill him if they get their claws on him. It is up to you to guide Percy around the ice safely and do away with the Snobees by pushing the ice blocks over them.

If you completely wipe the Snobees off the screen, then a new wave will appear. Do it fast enough and you will be awarded bonus points; apart from this, each round is the same as the previous one. The rounds are punctuated by an Electron rendition of an excerpt from a theme by Paganini - I am not familiar enough with this piece to know what the reason for this might be, but there is probably a hidden message there somewhere. While playing the game, there is an interminable cycle of Bach's Toccata so it's a shame that you can't turn the sound off from within the game. Percy's movements are a little slow to react to your instructions, but you quickly learn to adapt to this.

On the whole this is a rather nice game. It's quite addictive and I confess I like the little penguin character, he's very cute.

Title : Lunar Rescue
Supplier : Alligata
Price : £7.95 inc VAT
Rating : **
Reviewer : Alan Webster

Lunar Rescue is another Alligata game that has been transferred from the BBC to the Electron and like the others it doesn't quite play to the same standards. The game is slow on the first level, and never becomes any faster. After completing four levels without losing a life, I lost interest.

The object of the game is to navigate a spaceship through a force of alien ships and to land on the surface of a planet. Once you have landed, you pick up one of the six earthmen stranded on the planet, and fly him back to safety. The game is similar in concept to the 'Lunar Escape' game that we published in Elbug Vol.1 No.6.

Because of its lack of speed this is not one of the better games from Alligata, and the sound effects are not very stunning either.

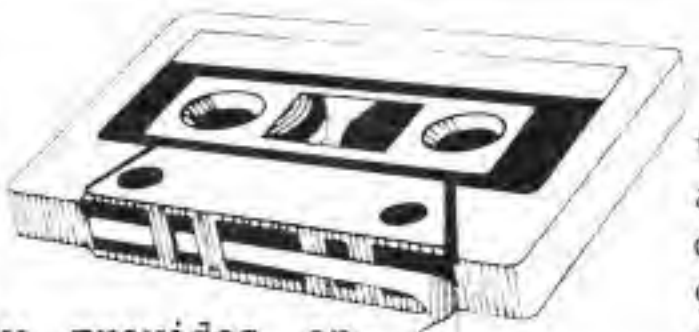
HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

USING TAPE RECORDERS WITH AUTOMATIC RECORD LEVEL CONTROL - B.Barrett

An automatic level controller on a tape recorder can be responsible for bad recordings of saved programs. If you have problems and if it is possible to do so on your recorder, this facility should be switched off.

ACCURATELY FILLING RECTANGULAR AREAS - M.Mertens

Filling a rectangular area on the screen using triangle plot options 80 to 87 can lead to the problem of unwanted lines when using the GCOL statement with first parameters greater than zero. This is caused by the desired plotting effect occurring twice at the same place. One method of filling a rectangular area which overcomes this problem is to define the area as a graphics window using VDU 24, setting the background colour and plot option to the required values and perform a CLG to fill the window. This is also a faster way to achieve the desired result. A second method involves plotting triangles as before, and drawing a line in the required colour and plot option where the triangles meet. (See 'Moving Chequer Board' in ELBUG Vol.1 No.4 lines 210 to 270.)



CASSETTE INDEXER

by

D.J. PILLING

If you have some difficulty managing all your programs saved on cassette then this handy utility will be a great help. It not only provides an index to programs that you have saved but it also works out the counter settings as well for speedy access.

One of the problems with keeping several programs on one cassette is that of finding the programs easily when you want to use them. Unless you know fairly accurately where a program is saved, much time can be wasted just finding where the start of the program is. Most cassette players used with computers have a counter which can be very useful in identifying the start of any given program. We now go one step further and automate the whole process of cataloguing a cassette of programs with the utility listed here.

The indexer utility will automatically read through a complete cassette of programs and produce an index giving for each program its name, its length in blocks and in bytes, starting time in seconds and starting counter reading. The indexer also has some capability to cope with programs which contain recording errors (i.e. which would not load under normal circumstances). If such a program is found, its length in bytes is entered as a blank, providing an indication of its bad state.

The rate at which the counter rotates varies (sometimes significantly), from one cassette recorder to another. For this reason, the indexer program contains a special calibration routine to enable it to work reasonably accurately with most cassette recorders. Once the program has been calibrated in this way (in effect by setting the values of two variables R and D) the indexer will automatically record the correct counter readings as it reads through a cassette of programs. In addition, the indexer program can also be used in manual mode, with you entering the correct counter value at the start of each program on a cassette, although this is unlikely to be as accurate. The indexer program contains full instructions to guide you through whichever option you choose.

THEORY

The program uses the two values assigned to the variables R and D to make its calculations. R is the initial effective radius of the right hand spool in the cassette, and D is the thickness of the tape. Usually, the tape counter is linked to the right hand spool. As the tape runs through the recorder the radius of the right hand spool gets larger as the tape is wound onto it. Thus it turns slower in order to maintain the same tape speed. This means that the number on the tape counter, is not literally related to the time elapsed since the start of the tape. The cause of the non-linearity is of course the finite thickness of the tape. The program will still work if the counter is linked to the left hand spool, but do not be worried if either R or D turns out to be negative - they are only theoretical values!

USING THE INDEXER PROGRAM

Before using the indexer program you should make sure you have at least one copy securely saved. This is important as the indexer program modifies itself when it runs, leaving you only with the cassette index at the end of its run. The program details the three choices open to you. These are:

1. Manual indexing of cassette.
2. Automatic indexing of cassette.
3. Calibration and indexing.

This program was originally designed for cassette recorders that had remote motor control. In order that the

program may also be used with other cassette recorders, a separate calibration program is provided for use in this situation. See the end of the article for further details on this.

AUTOMATIC CALIBRATION

This section allows the values of R and D for your cassette recorder to be calculated precisely. Once selected, the tape will start running, and you should press the space bar when the prompt is displayed. You will then be asked to enter the counter reading. The program will continue to read through the tape, indexing the programs as it finds them. After a reasonable interval (to get a good average reading) press the space bar a second time (when prompted) and enter the new counter reading. The program then calculates the correct values of the constants R and D before continuing to index the programs on the tape as described below. Do make sure that the counter is reset to zero before starting any calibration or indexing.

AUTOMATIC INDEXING

With this option, the program reads through the tape, and notes down each file, its length, where on the tape it starts, and at what tape counter value. When you are satisfied that all the files have been included, press Escape ONCE ONLY. The program will then modify itself, leaving you with a small program that when run, lists out the files on the tape, in order, with their associated statistics. You can then save this program if required, as a permanent record.

MANUAL INDEXING

With this option, you are required to enter the value of the tape counter at the start of each program, but aside from that it is functionally equivalent to the automatic indexing option. Because of the way in which this option is performed, only cassette users with motor control will be able to use this option correctly.

```
  1 REM PROGRAM TAPE INDEXER
  2 REM AUTHOR  D.J. PILLING
  3 REM VERSION E1.1
  4 REM ELBUG   JULY 1984
  5 REM PROGRAM SUBJECT TO COPYRIGHT
  6 :
 50 MODE6:@%=&20406:OSBYTE=&FFF4
 60 PROCINS
 70 *OPT1,1
 80 DIMN$(100),T$(100),C$(100)
 90 ONERROR GOTO580
 95 PROCTITLE
100 PRINT''''"(1) MANUAL""(2) AUTO"'
"(3) CALIBRATE"''"SELECT TAPE COUNTER
MODE: ";:INPUT""C%
105 VDU19,0,0,0,0,0,19,1,3,0,0,0
110 IFC%<1 OR C%>3 GOTO100
120 TS%=0:TF%=0:N%=0
130 REM ============================
140 REM DEFAULT R AND D
150     R=.2779:D=.0003
160 REM ============================
170 CLS:TIME=TF%:ER%=FALSE
180 X%=OPENUP("")
190 TS%=TIME:N%=N%+1:PROCCAL:TIME=TS%
200 REPEAT Z%=BGET# X%:UNTIL EOF#X%
210 CLOSE #X%
220 TF%=TIME
230 PROCST
240 GOTO170
250 IF C%>2 PROCCX
260 HIMEM=HIMEM-8000:P%=HIMEM
270 PROCDS:P%=HIMEM
280 L%=6000
290 F$=STR$L%+"D."+STR$N%+CHR$13+"G.3
10"+CHR$13
300 PROCWR(F$)
310 L%=6010:I%=1
320 F$=STR$L%+"D."+FNSS+CHR$13+"G.340
"+CHR$13
330 PROCWR(F$)
340 L%=L%+5
350 F$=STR$L%+"D."+FNSS+","+FNSS+CHR$
13+"G.370"+CHR$13
360 PROCWR(F$)
370 L%=L%+10:I%=I%+1
380 IFI%>N% ELSE GOTO320
390 F$="DELETE0,1500"+CHR$13
400 CLS:PRINT'''"I'M YOUR INDEX"''"SAV
E ME"
410 PRINT''''"(The program in the comp
uter is now an"'"index of the tape; whi
ch you may run"'"and save.)"''
420 PROCWR(F$)
430 END
440 DEFPROCWR(A$)
450 FORJ%=1TOLENA$
460 A%=138:X%=0:Y%=ASC(MID$(A$,J%,1))
470 CALL&FFF4
480 NEXT:END
490 DEFPROCDS
500 FORI%=1TON%
510 $P%=N$(I%):P%=P%+LENN$(I%)+1
520 $P%=T$(I%):P%=P%+LENT$(I%)+1
530 $P%=C$(I%):P%=P%+LENC$(I%)+1
```

```
  540 NEXT:ENDPROC
  550 DEFFNSS
  560 A$=$P%:P%=P%+LEN$P%+1
  570 =A$
  580 IF ERR=17 AND ERL=180 GOTO250
  590 IF ERR=17 AND ERL=200 N%=N%-1:GOT
O250
  600 IF ERR>200 TF%=TIME:PRINT:ER%=TRU
E:PROCST:GOTO170
  610 ONERROROFF:REPORT:PRINT" at line
";ERL:END
  620 DEFPROCST
  630 F$=FNS(0,VPOS-1)
  640 L$=RIGHT$(F$,4)
  645 IFER% L$="    "
  650 B$=MID$(F$,12,2)
  660 F$=LEFT$(F$,10)
  670 IFMID$(F$,10,1)<>" " GOTO700
  680 REPEAT F$=LEFT$(F$,LENF$-1)
  690 UNTIL MID$(F$,LENF$,1)<>" "
  700 N$(N%)="""+F$+B$+L$+"""
  710 T$(N%)=STR$(TS%DIV100)
  720 C$(N%)=C$
  725 ENDPROC
  730 DEFFNS(X,Y)
  735 OX%=POS:OY%=VPOS
  740 A%=135:A$=""
  745 FORI%=0TO17:VDU31,(X+I%),Y
  750 CH$=CHR$((USR(OSBYTE)AND&FFFF)DIV
256)
  755 A$=A$+CH$:NEXT
  760 VDU31,OX%,OY%
  765 =A$
  770 DEFPROCCAL
  780 *FX21,0
  790 Y%=VPOS
  800 IF C%=1:VDU7,7:INPUTTAB(0,Y%+1)"N
UMBER ON TAPE COUNTER  "C$:IFC$=""GOTO8
00
  810 IF C%=2 C$=FNTC(TS%DIV100)
  820 IF C%>2 C$="":PROCCALS
  830 PRINTTAB(0,Y%);
  840 ENDPROC
  850 DEFPROCCALS
  860 VDU7,7
  870 PRINTTAB(0,Y%+1)"TO GO FOR CALIBR
ATION PRESS SPACE BAR   ELSE PRESS ANY
KEY TO CONTINUE":IK=GET
  880 IF IK<>32 PRINTTAB(0,Y%+1)SPC(79)
:ENDPROC
  890 INPUT'"INPUT NUMBER ON TAPE COUNT
ER  "N1%
  900 IFN1%=0 PRINT"ERROR ZERO ENTERED"
:IK=INKEY(200):ENDPROC
  910 IF C%=3 N2%=N1%:T2%=TS%DIV100:C%=
4:ENDPROC
  920 IF N1%=N2% PRINT"ERROR SAME NUMBE
R ENTERED TWICE":IK=INKEY(200):GOTO890
  930 T1%=TS%DIV100:PROCCDR
  940 IFD=0 PRINT"ERROR INSUFFICIENT AC
CURACY"'"TRY AGAIN LATER":IK=INKEY(200)
:ENDPROC
  950 PROCCX:C%=2:C$=FNTC(TS%DIV100)
  960 PRINT'"CALIBRATION COMPLETED"'
  970 PRINT"YOUR D VALUE = ";D
  980 PRINT"YOUR R VALUE = ";R
  990 PRINT'"(Retain all digits. Next t
ime you LOAD"'"the program enter R and
D in Line 150.)"
 1000 PRINT'"ANY KEY TO CONTINUE"
 1010 IK=GET
 1020 ENDPROC
 1030 DEFPROCCDR
 1040 D=(T1%/N1%-T2%/N2%)/PI/(N1%-N2%)
 1050 R=(T1%/N1%/(N1%-1)-T2%/N2%/(N2%-1
))/PI/2/(N2%-N1%)*(N1%-1)*(N2%-1)
 1060 ENDPROC
 1070 DEFFNTC(ST%)
 1080 =STR$(INT((D-2*R+SQR((2*R-D)^2+4*
ST%*D/PI))/2/D+0.5))
 1090 DEFPROCCX
 1100 FOR J%=1TON%:C$(J%)=FNTC(VAL(T$(J
%))):NEXT:ENDPROC
 1110 DEFPROCINS
 1115 VDU19,0,4,0,0,0:VDU19,1,3,0,0,0
 1120 PROCTITLE
 1125 VDU7:PRINT'"WARNING THIS PROGRAM
 IS SELF"'"MODIFYING. BEFORE PROCEEDING
"'"MAKE SURE YOU HAVE A COPY."'TAB(17)
" ANY KEY TO START";:IK=GET:CLS:*FX21,0
 1130 PROCTITLE:VDU19,1,7,0,0,0,19,0,4,
0,0,0
 1135 PRINT'"HOW TO USE THE PROGRAM"'"P
ut the tape you want an index of in"'"t
he recorder. Zero the tape counter."'"
You now have three options:"
 1140 PRINT"(1) MANUAL"'"Enter the numb
ers on the counter"'"at the start of ea
ch program as"'"prompted."
 1150 PRINT"(2) AUTO"'"Let the program
calculate the"'"tape counter readings;
using default"'"R and D values previous
ly determined"'"for your recorder and t
ype of tape"'"and entered in line 150."
 1160 PRINT"(3) CALIBRATE"'"Input two t
ape counter readings to"'"calibrate the
 program for your"'"recorder and then p
roceed as in (2)."
 1180 PRINT''TAB(17)"SHIFT TO START";
 1190 REPEAT UNTILINKEY-1:*FX21,0
 1200 ENDPROC
 1210 :
 1220 DEF PROCTITLE
 1230 CLS
 1240 PRINTTAB(13);"TAPE INDEXER  ";
 1250 ENDPROC
 1260 :
 5000 CLS:VDU19,1,3,0,0,0:@%=&20106
 5005 PRINTTAB(17)"TAPE INDEX  "
```

```
 5010 Y%=1:RESTORE6000:READN%:PRINT"NAM
E"TAB(12)"B"TAB(16)"L"TAB(23)"TIME"TAB(
31)"TAPE"
 5020 FORI%=1TON%
 5030 READN$,T%,C%
 5040 L$=RIGHT$(N$,4):B$=MID$(N$,LENN$-
5,2):N$=LEFT$(N$,LENN$-6)
 5050 PRINTN$TAB(12)B$TAB(16)L$TAB(23);
T%TAB(31);C%
 5060 NEXT
```

NOTE: The program above uses some rather unconventional coding techniques, which also accounts for the fact that the line numbers do not follow our usual pattern. Do not renumber this program.

MANUAL CALIBRATION

Those whose cassette recorders do not have motor control should use this separate program to calibrate the system before using option 2 in the main program already described. When you run this calibration program, switch your cassette recorder to 'play' and note the counter reading on each of the two occasions that you hear a bleep. The program will then ask you to enter these two readings which are used to calculate the values of R and D, and these should then be included in the main indexer program at line 150.

```
  10 REM PROGRAM NO MOTOR CALIBRATE
  20 REM AUTHORS D.J. PILLING/ D. FELL
  30 REM VERSION E1.0
  40 REM ELBUG   JULY 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 *OPT1 1
 110 MODE6
 120 VDU19,1,3,0,0,0
 130 @%=&20406
 140 PRINT"MANUAL CALIBRATION."
 150 PRINT'"REWIND TAPE TO START"'"ZER
O TAPE COUNTER"
 160 REPEATUNTILINKEY(100)=TRUE
 170 PRINT'"O.K. ?"
 180 PRINT'"SWITCH TO PLAY AND HIT RET
URN"
 190 REPEATUNTILGET=13
 200 PRINT'"NOTE THE COUNTER VALUES O
N THE TONES."
 210 TIME=0
 220 X%=OPENUP("")
 230 T2%=TIME DIV 100
 240 SOUND17,-15,100,10
 250 REPEAT B%=BGET#X%:UNTILEOF#X%
 260 SOUND17,-15,100,10
 270 T1%=TIME DIV 100
 280 CLOSE#X%
 290 INPUT"FIRST COUNTER VALUE "N2%'"S
ECOND COUNTER VALUE "N1%
 300 PROCCDR
 310 PRINT"D VALUE IS "D'"R VALUE IS "R
 320 PRINT"NOTE THESE AND USE AT LINE
150 OFTHE"'"MAIN PROGRAM. ALL DIGITS AR
E SIGNIFICANT"
 330 END
 340 :
1000 DEFPROCCDR
1010 D=(T1%/N1%-T2%/N2%)/PI/(N1%-N2%)
1020 R=(T1%/N1%/(N1%-1)-T2%/N2%/(N2%-1
))/PI/2/(N2%-N1%)*(N1%-1)*(N2%-1)
1030 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

STRANGE CRASH - R.Tobin

Type G.0:G.0:G.0:G.0:G.0:....etc., until you've filled the buffer (about 6 lines in Mode 6). Then hit return. You will of course get the error 'No such line' but it also has other effects...

TAB COMMAND - R.Sterry

When using the short, TAB(x) form of the TAB instruction (User guide p.188) to print further along the line, you may sometimes find that some text to the left of the printing position will be deleted. For example

```
10 PRINT"hello":pause=INKEY(200):VDU11
20 PRINT TAB(20)"there"
```

The TAB(x,y) form of the instruction can be used to remedy this using the VPOS Basic system variable. VPOS is used by Basic to keep a record of which row on the screen is being used for printing. So if you find this problem in a program, then replace the TAB(x) with TAB(x,VPOS). In the example, line 20 would become:

```
20 PRINT TAB(20,VPOS)"there"
```

ELECTRON BREAKOUT

by Alan Webster

Breakout is one of the classic computer games. Although other games may have more sophisticated graphics displays, this remains one of the simplest yet most addictive of computer games.

The game of Breakout is very easy to understand, but nowhere as easy to play. A wall comprising several rows of colourful bricks is displayed across the top of the screen. You control a bat which you move left and right across the bottom of the screen, repeatedly hitting a bouncing ball at the wall, to demolish the bricks. The bounce is somewhat unpredictable and at times confusion reigns as the ball rebounds from several bricks and the surrounding frame in rapid succession. Break through the wall however, into the space above, and the bouncing ball will really go bananas! The ball can be 'steered' from the bat, by moving the bat in the direction you wish the ball to travel as it hits the bat.

At the start of a game you can select the degree of difficulty that you want to face - the degree of difficulty selected controls the size of your bat. To move the bat you use the usual Z (for left) and X (for right) keys. Complete the demolition of the whole wall and a wall of a new colour appears, slightly lower on the screen so that your reactions will need to get faster and faster. At the end of a wall you will get bonus points according to the time taken to demolish the wall.

The game has the merit of being quite short, compared with many that we publish; it is good fun, and well worth playing.

PROGRAM NOTES

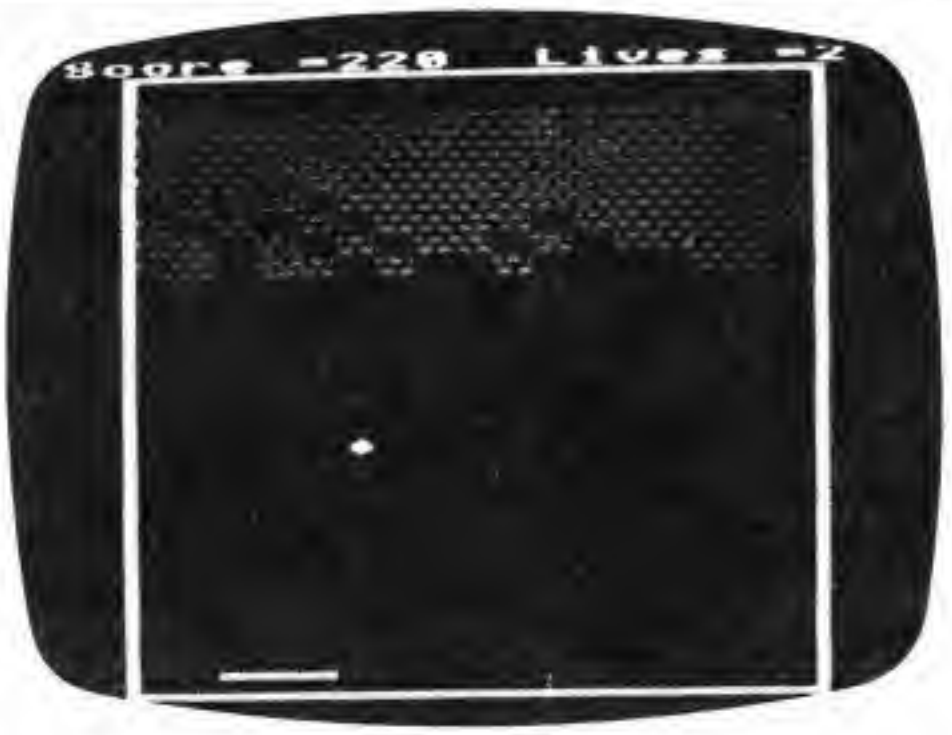
It is quite difficult to write a fast moving action game in Basic on the Electron. To help, this program includes some short sections of machine code and direct references to memory locations to speed the game up. We shall be telling you more about these

techniques in future issues of Elbug. Take care when typing the program in, particularly with these sections (1130-1240 and 1270-1450). You should save the program before attempting to run it, as you would with any program that contains machine code.

```
  10 REM Program BREAKOUT
  20 REM Version E0.7
  30 REM Author  Alan Webster
  40 REM ELBUG   July 1984
  50 REM Program Subject to Copyright
  60 :
 100 ON ERROR GOTO 1530
 110 MODE1
 120 PROCchars
 130 PROCinstr
 140 MODE5
 150 PROCmain
 160 IF FNend=1 GOTO 110
 170 MODE6
 180 END
 190 :
1000 DEFPROCchars
1010 VDU23,224,255,255,255,0,0,0,0,0
1020 VDU23,225,0,0,0,12,30,30,30,12
1030 VDU23,227,0,221,221,221,0,119,119
,119
1040 VDU23,228,7,7,7,7,7,7,7,7
```

→

```
 1050 VDU23,229,224,224,224,224,224,224
,224,224
 1060 ENDPROC
 1070 :
 1080 DEFPROCassemble
 1090 V=&FFEE
 1100 DIM space 50
 1110 FOR h%=0 TO 2 STEP2
 1120 P%=space
 1130 [OPT h%
 1140 .screen
 1150 LDA #17:JSR V
 1160 LDA #1 :JSR V
 1170 LDY &7F:.wall LDX #18:.wall2
 1180 LDA #31:JSR V:TXA:JSR V
 1190 TYA:JSR V:LDA #227:JSR V
 1200 DEX:CPX#1:BNE wall2
 1210 DEY:CPY&80:BNE wall
 1220 RTS
 1230 .chr LDA#135:JSR &FFF4:STX &75:RTS
 1240 ]:NEXT
 1250 ENDPROC
 1260 :
 1270 DEFPROCgame
 1280 ?&74=?&70
 1290 ?&70=?&70+(INKEY-98 AND ?&70>2)-(
INKEY-67 AND ?&70<(19-BL%))
 1300 COLOUR2
 1310 IF?&70=?&74GOTO1330
 1320 VDU31,?&74,29:PRINTSTRING$(BL%,CH
R$32)
 1330 VDU31,?&70,29:PRINTSTRING$(BL%,CH
R$224)
 1340 x=?&71:y=?&72
 1350 COLOUR3
 1360 ?&71=?&71+mx:IF ?&71<2 OR ?&71>18
 ?&71=?&71-mx:mx=-mx:SOUND3,-10,2,2
 1370 ?&72=?&72+my:IF ?&72<2 ?&72=?&72-
my:my=-my:SOUND2,-10,2,2
 1380 VDU31,?&71,?&72
 1390 CALL chr
 1400 IF ?&72>29 AND ?&75=32 LV%=LV%-1:
lost=TRUE:FORA%=100 TO0 STEP-10:SOUND2,
-10,A%,1:NEXT:ENDPROC
 1410 PROCbrick
 1420 IF ?&72=29 AND ?&75=128 ?&72=28:m
y=-my:SOUND1,-10,50,2:PROCguide
 1430 VDU31,?&71,?&72,225
 1440 VDU31,x,y,32
 1450 ENDPROC
 1460 :
 1470 DEFFNend
 1480 COLOUR128:COLOUR1
 1490 PRINTTAB(5,13)"Game Over."
 1500 PRINTTAB(5,15)"New Game? ";CHR$12
7;
 1510 INPUT""Y$:IF Y$="Y" =1 ELSE =0
 1520 :
 1530 ON ERROR OFF
 1540 MODE 7
```




```
 1550 IF ERR=17 END
 1560 REPORT:PRINT" at line ";ERL
 1570 END
 1580 :
 1590 DEFPROCmain
 1600 VDU19,3,6,0,0,0
 1610 CLS
 1620 SC%=0:LV%=3
 1630 C=0
 1640 VDU23,1,0;0;0;0;
 1650 PROCassemble
 1660 REPEAT
 1670 VDU19,1,0,0,0,0
 1680 COLOUR128
 1690 CALL screen
 1700 VDU19,1,(C MOD 7)+1,0,0,0
 1710 S%=0
 1720 REPEAT
 1730 TIME=0
 1740 won=FALSE:lost=FALSE
 1750 ?&70=9
 1760 ?&71=RND(14)+2
 1770 mx=1:my=-1
 1780 ?&72=RND(25-?&7F)+?&7F
 1790 COLOUR2
 1800 PRINTTAB(1,30)STRING$(19,CHR$224);
 1810 PRINTTAB(1,1)STRING$(19,CHR$224);
 1820 FORW%=1 TO30:PRINTTAB(1,W%);CHR$2
28;TAB(19,W%)CHR$229;
 1830 NEXT
 1840 REPEAT
 1850 PROCgame
 1860 PRINTTAB(0,0);"Score=";SC%
 1870 PRINTTAB(12,0);"Lives=";LV%
 1880 IF S%=1360 won=TRUE:S%=0
 1890 UNTIL lost OR won
 1900 VDU31,?&70,29:PRINTSTRING$(BL%,CH
R$32)
 1910 UNTIL LV%<1 OR won
 1920 IF won C=C+1:VDU31,?&71,?&72,32
 1930 ?&80=?&80+1:?&7F=?&7F+1
 1940 IF 1000-(TIME/100)<0 GOTO1960
```

→

```
 1950 IF won PRINTTAB(4,12)"BONUS =>";:
B%=1000-(TIME/100):PRINTTAB(13,12);B%:S
C%=SC%+B%
 1960 IF won FORA=1TO4000:NEXT:PRINTTAB
(4,12)STRING$(13,CHR$32):UNTIL LV%<1
 1970 *FX15
 1980 ENDPROC
 1990 :
 2000 DEFPROCinstr
 2010 VDU19,3,6,0,0,0
 2020 PRINT'''
 2030 COLOUR2:PRINTTAB(16)"BREAKOUT":CO
LOUR1:COLOUR130:PRINTTAB(15,VPOS)STRING
$(10,CHR$227)
 2040 COLOUR2:COLOUR129:PRINTTAB(12,VPO
S+2)"by  Alan Webster"
 2050 PRINT''':COLOUR3:COLOUR128
 2060 PRINT" The object of this game  i
s to demolishthe brick wall by bouncing
 a ball at it.When you have destroyed
a wall,  you goon to the next one. You
 score 10 pointsfor each brick  that yo
u  knock out  andyou start with three l
ives."
 2070 PRINT'" The following keys are us
ed for moving:"
 2080 PRINTTAB(10)"Z - Bat Left"'TAB(10
)"X - Bat Right"
 2090 PRINT'''"Please enter skill level
- 1 to 7 (level1 is easy, level 7 is ha
rd):";
 2100 REPEAT:BL%=GET-48:UNTIL BL%>0 AND
 BL%<8:VDU BL%+48:BL%=8-BL%
 2110 PRINT''''"Press any key to start:"
;:G=GET
 2120 ?&80=2:?&7F=10
 2130 ENDPROC
 2140 :
 2150 DEFPROCbrick
 2160 IF NOT(?&75=131 OR ?&75=128) ENDP
ROC
 2170 IF ?&75=131 SC%=SC%+10:S%=S%+10
 2180 IF ?&75=131 SOUND&10,-4,100,4:my=
-my
 2190 mx=SGN(mx)
 2200 IFmx=0 mx=-1
 2210 mx=SGN(mx)*(RND(2)-1)
 2220 IF RND(2)=1 mx=-mx
 2230 ENDPROC
 2240 :
 2250 DEFPROCguide
 2260 G=0
 2270 IF INKEY-67 mx=1:G=1
 2280 IF INKEY-98 mx=-1:G=1
 2290 IF G=0 PROCbrick
 2300 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SHORTENED 'IF' STATEMENT - R.Sterry

The IF-THEN statement can be shortened by leaving out the THEN keyword. For example

```
100 IF animal%=5 THEN PRINT "It's the wolf!"
```

can be shortened to

```
100 IF animal%=5 PRINT "It's the wolf!"
```

which is not quite so clear as the former but may prove to be convenient on some occasions.

ERROR REPORTING AND EDITING - R.Sterry

When this short error trapping routine (lines 10000 to 10040) is added to your Basic program, it will only take effect if your program fails with an error, but it will return the computer to you in a controlled manner and display the program line that caused the program to fail.

```
   10 ON ERROR GOTO 10000
  100 nonsense nonsense nonsense : REM this is nonsense for Basic
10000 MODE 6:IF ERR=17 THEN 10050
10010 REPORT:PRINT" at line ";ERL:*FX15,0
10020 err$="L."+STR$(ERL)+CHR$(11)+CHR$(13)
10030 FOR bit%=1 TO LEN(err$)
10040 OSCLI("FX138,0,"+STR$(ASC(MID$(err$,bit%,1)))):NEXT
10050 END
```

Line 10 initiates an error trapping sequence so that lines 10000 onwards are executed in the event of an error in Basic. Line 100 is included so that if you run the short program as listed, you will see how this works.

ELECTRON GRAPHICS (Part 8)

by Mike Williams

In our continuing series on Electron graphics we turn our attention to some useful and interesting techniques for including colour in screen displays.

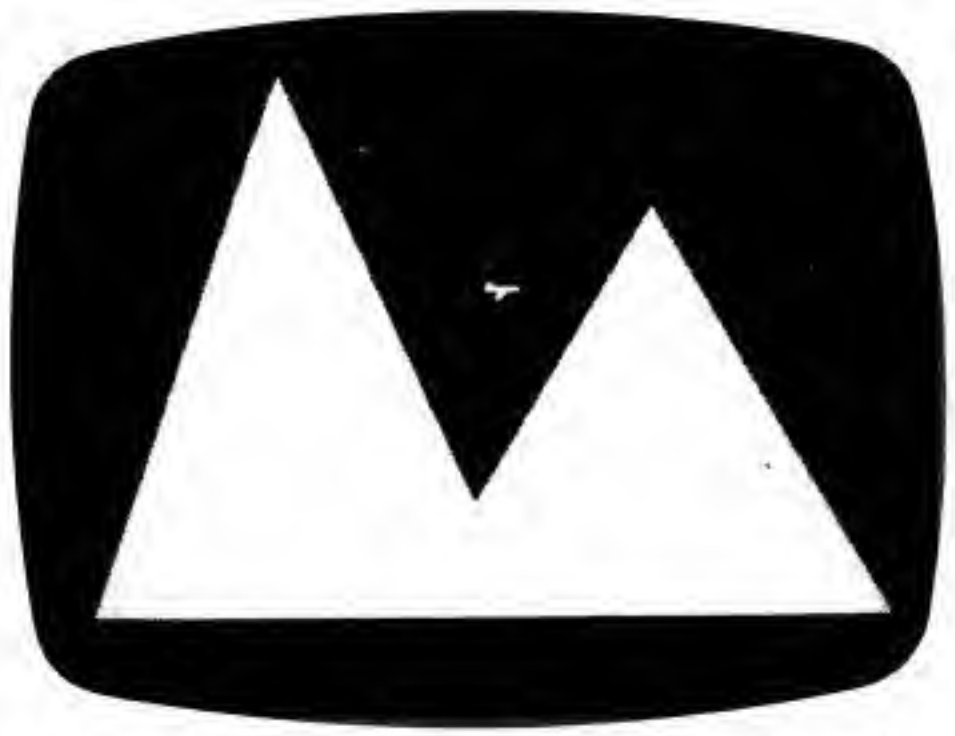
In this month's article we shall be concentrating on the use of colour when drawing in any of the graphics modes. The number of colours available depends on the mode chosen and is related also to the resolution of the graphics screen as was described in the March issue of ELBUG. In our examples we will concentrate on the two four-colour modes (modes 1 and 5).

In a four colour mode, the four colours are black, red, yellow and white (though these can be changed as we shall see later), and these colours are referred to by the numbers 0, 1, 2 and 3. Even if we do change the actual colours themselves, these same four numbers are always used to specify colours in modes 1 and 5. In both text and graphics we can define a foreground and a background colour. The foreground colour is a number in the range 0 to 3, while a background colour has 128 added to the colour number.

In graphics, which is what we are principally concerned with here, the GCOL instruction is used to specify what colour we want to use at any stage of our program. For example if we start a program with the lines:

```
100 MODE 1
110 GCOL 0,1
120 GCOL 0,130
130 CLG
```

then after selecting mode 1, the instructions specify red (colour 1) for the foreground colour, and 130 (i.e.128+2) for the background colour, which is thus yellow. Remember that a GCOL instruction never, by itself, changes the colour of anything already visible on the screen. Hence the CLG



instruction at line 130 which actually paints the whole screen area in the background colour of yellow.

The GCOL instruction always requires the use of two numbers. The second is the colour as described above, while the first determines the mode of plotting that will take place. This number will normally be in the range of 0 to 4. The value of 0 used above means that any graphics will be drawn in the colour specified regardless of any other colours already existing in the same area of the screen.

You will have seen the use of mode 3 (or Exclusive-OR) plotting in some of the early articles in this series about moving characters across the screen. Using this technique, any shape or character can be made to move on the screen by drawing the shape, then drawing the shape a second time in the same position before repeating the process in a new position. The use of Exclusive-OR plotting means that when a shape is plotted in the same position a second time, it effectively disappears

from the screen, and the background is restored to its original colour.

The only disadvantage of Exclusive-OR plotting is that an object appears to change colour when it is plotted over a background other than black. This article will show you how to control this to produce even more interesting effects.

Let's now write a program that we can use to demonstrate some new ideas. The main idea of this program is to show a plane flying across a mountainous landscape.

```
  10 REM Program PLANE
  20 REM Version E1.0
  30 REM Author Mike Williams
  40 REM ELBUG July 1984
  50 REM Program subject to copyright
  60 :
 100 MODE 1
 105 ON ERROR GOTO 190
 110 PROCcolours
 120 PROCdefplane
 130 PROCmountains
 140 REPEAT
 150 PROCflyplane
 160 UNTIL FALSE
 170 END
 180 :
 190 ON ERROR OFF:MODE 6
 200 IF ERR<>17 REPORT:PRINT" at line
";ERL
 210 END
 220 :
1000 DEF PROCcolours
1080 ENDPROC
1090 :
1100 DEF PROCdefplane
1110 VDU23,224,0,48,120,127,127,3,7,7
1120 VDU23,225,0,0,0,254,255,192,128,0
1130 plane$=CHR$224+CHR$225
1140 ENDPROC
1150 :
1200 DEF PROCmountains
1210 GCOL0,1
1220 MOVE 0,0:MOVE 300,800:PLOT85,700,0
1230 MOVE 500,0:MOVE 900,600:PLOT85,128
0,0
1240 ENDPROC
1250 :
1300 DEF PROCflyplane
1310 GCOL3,2:VDU5
1320  FOR I%=0 TO 1216 STEP 8
1330 PROCplane(I%,500)
1340 NEXT I%
1350 VDU4
1360 ENDPROC
1370 :
1400 DEF PROCplane(xpos,height)
1410 MOVE xpos,height:PRINT plane$
1420 Z%=INKEY(1)
1430 MOVE xpos,height:PRINT plane$
1440 ENDPROC
```

The main program is quite short. The procedure called PROCcolours will be filled in later. The procedure PROCdefplane defines new characters (ASCII 224 and 225) that together form the crude outline of a plane. For convenience the two character string is assigned to the variable plane$, just as described in parts 2 and 3 of this series (issues 2 and 3 of ELBUG).

The procedure PROCmountains draws two mountains on the screen using PLOT85, the triangle plotting instruction that we looked at in issue 5. This procedure also selects colour 1 (red) for the mountains.

The next part of the program is a loop that is repeated indefinitely. Each repetition calls the procedure PROCflyplane to fly the plane across the screen. This procedure itself is basically a FOR-NEXT loop which in turn calls the further procedure PROCplane to plot the plane in a series of positions across the screen. Notice the step size of 8, to achieve reasonable movement across the screen. Larger step sizes will produce faster but jerkier movement. The minimum step size in this mode to achieve real movement is 4 (see the graphics article in issue 6 of ELBUG). The final value in the FOR loop is the last position in which the whole plane will be visible on the screen (each character in mode 1 is equivalent to 32 graphics units - i.e.1280/40), and avoids any complications with part planes.

The procedure PROCflyplane also selects the colour (yellow) and Exclusive-OR plotting for the drawing of the plane. It also uses VDU5 so that the two characters that form the plane can be plotted at the graphics cursor position (this is restored to normal with a VDU4 at the end of the

procedure). The procedure PROCplane, which actually displays the plane on the screen, is called with two parameters, the horizontal and vertical position of the plane on the screen in graphics co-ordinates.

Lastly, the procedure PROCplane displays the plane on the screen for a short time before overwriting it in the same position to make the plane disappear. This gives the slightly flickering effect as the plane is first drawn and then erased as it moves across the screen. Changing the value of the INKEY statement will also affect the apparent speed of movement of the plane.

If you run the program as listed above, you should repeatedly see the plane move across the screen from left to right. Unfortunately, the yellow plane changes to white as it moves across the mountains.

To see why, we will now look in detail at what happens with colours when we use Exclusive-OR plotting. We shall then be able to change our program slightly so that the plane remains the same colour, either moving in front of the mountains, or behind them as we choose.

First of all we need to write down the four colour numbers in binary:

black	Ø	ØØ
red	1	Ø1
yellow	2	1Ø
white	3	11

The Exclusive-OR operation compares the bits (binary digits) of the two colour numbers together (foreground and background colours), and produces a further binary number which determines the colour we see. If the two binary digits are different then a '1' results, if they are the same then a 'Ø' results.

Now the plane is yellow (colour 2) and the mountains are red (colour 1). Comparing them together and performing an Exclusive-OR operation gives the following results:

yellow ⟶ 1Ø
red ⟶ Ø1
11 ⟶ white

In each case the corresponding binary digits are different and a '1' results in both positions. This is the binary for 3 which is white, the colour we see for the plane as it moves across the mountains.

Now if only colour 3 were not white but red, then the colour of the plane as it crosses the mountains would make it invisible, and it would in fact appear to move behind the mountains on the screen. In fact it is quite easy to do this using the VDU19 instruction. This allows us to specify any colour number (for the mode we are in), and the actual colour (out of the 16 available flashing and non-flashing colours) that we would like it to be. The two numbers must always be followed by three zeros as in the examples. All the 16 colours and their corresponding numbers are as specified for mode 2 in the User Guide on page 141. In this case, add the following line to the program as part of the procedure PROCcolours:

```
1Ø1Ø VDU19,3,1,Ø,Ø,Ø
```

This changes colour 3 (in mode 1) to colour 1 which is red. If you enter this line and re-run the program you should see the desired effect. Alternatively, use this version:

```
1Ø1Ø VDU19,3,3,Ø,Ø,Ø
```

This time colour 3 (out of the 4 in mode 1) is changed to colour 3 (out of 16), and thus the plane remains yellow throughout. The VDU 19 instruction is very useful as it means that in a mode like mode 1 where only four colours are available, we can choose which four colours we want to use. We don't have to stay with the default colours of black, red, yellow and white.

To see this, add the following four lines to the program to make up the procedure PROCcolours:

→

```
1010 VDU19,0,6,0,0,0
1020 VDU19,1,2,0,0,0
1030 VDU19,2,4,0,0,0
1040 VDU19,3,2,0,0,0
```

This has the effect of changing black to cyan (for the sky), red to green (for the mountains), yellow to blue (for the plane), and white to green (so that the plane appears to fly behind the mountains).

The VDU19 instruction is very useful in allowing you to choose the colours that you to see on the screen. Be warned, however, as unlike the GCOL instruction, VDU19 can have an immediate effect on the screen display. For example, changing red into blue will result in all areas or lines in red on the screen changing immediately into blue. You are still limited by the mode you use as to how many colours you can have on the screen at any one time, but at least you can now choose which those will be.

As we have seen, four colour modes (modes 1 and 5) allow you to have not only a foreground (the mountains) and a background (the sky), but also an apparent middle ground (occupied by the plane). The 16 colour mode 2 allows you to have even more layers to add depth to your pictures. I'll leave experimenting with this mode to you. Next month we will have a look at some more interesting things we can do with colours on the Electron.

POINTS ARISING

DOMINOES GAME (ELBUG Vol.1 No.5)

We have had some correspondence regarding this game published in the April issue of ELBUG. Because of the memory space required by this program, the coding had been 'squeezed', to remove any unnecessary spaces. However, this can only be done after a program has been entered, and as a result two lines in the program will cause problems if entered as listed. The correct versions (and the spaces are important) are:

```
 330 IF ELK% PROCELK ELSE PROCdisphand(1,0):PROCplayer
1470 IF bd%>pd% ELK%=TRUE ELSE ELK%=FALSE:s$=n$
```

You should also note that there should be a final closing bracket at the end of line 1670 (this had slipped slightly in production but was otherwise quite clear) while the fourth variable at line 2430 should be 'j%' and not just 'j'. Because this is in a LOCAL statement this small error causes no problem at all.

The version on the magazine cassette is quite correct apart from the inconsequential error in line 2430.

This month's issue of ELBUG sees the start of a short series on the problems of debugging programs, which should help you to sort out problems like these described above when they occur in your own programs, as well as in the magazine.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

CLEARING SCREENS WITH WINDOWS - K.Allen

With no graphics or text windows defined (see the VDU24 and VDU28 commands on pages 110 and 112 of the User Guide), the two commands CLG and CLS have the same effect, i.e. clearing the screen. However, if you define a graphics and/or a text window, CLS will only clear the text window and CLG will only clear the graphics window. If only a graphics window is defined, then CLG will clear this window, but CLS will clear the whole screen. Similarly, with only a text window defined, CLS will clear this window and CLG clears the whole screen. The reason for this is because CLG and CLS each clears the whole graphics or text window respectively. If any part of one window overlaps the other, then it will be cleared by either of the clear commands. Initially, both text and graphics windows are set to the whole screen area by default.

SCRAMBLE YOUR SOFTWARE - PROGRAM PROTECTION

by P Boardman

This interesting little utility is for those of you who wish to protect your own programs, making them unusable by anyone who does not know a prearranged password. It comes in the form of a procedure which must be added on to the end of the Basic program to be protected.

To use it, first type the procedure in and save it away on cassette. Then 'spool' it out to cassette under a different name. Spooling is achieved as follows:

```
*SPOOL LOCKSP   <return>
Set recorder to 'Record'
LIST            <return>
*SPOOL          <return>
Stop recorder.
```

This now means that you have a version of the program saved, ready to be *EXECed on to the end of your program to be protected (see the article on the use of functions and procedures in issue 5 of ELBUG for a full explanation of *SPOOL and *EXEC when used to merge programs. See also the User Guide page 200).

```
  100 REM EXAMPLE PROGRAM
  110 PRINT "THIS PROGRAM IS UNLOCKED"
  120 END
32000  DEFPROCLOCK
32010  INPUT"PASSWORD ",KEY$
32020  I%=PAGE+3
32030  REPEAT
32040  FORX%=I%+1 TO I%-4+?I%
32050  Y%=&1F AND (ASC MID$(KEY$,X% MOD
 LEN KEY$,1))
32060  ?X%=(?X% AND &E0)+((?X% AND &1F)
 EOR Y%)
32070  NEXT
32080  I%=X%+3
32090  UNTILX%>TOP-&D0
32100  CLEAR
32110  ENDPROC
```

Once that has been done, load in your Basic program making sure that no line numbers exceed 32000. Then type:

```
*EXEC LOCKSP <return>
```

and wait for the spooled version of the protection routine to be loaded on to the end of your program ignoring any error messages that may occur.

To scramble the program, type PROCLOCK, and type in a password when prompted. The password may contain any characters and be up to 256 characters in length, but you will have to remember the password you choose, so keep it short and meaningful. Once the password has been entered, the machine will pause, while it scrambles your program, changing the bytes of code and effectively locking your program. When that is completed, LIST your program and look at the code - it should make horrific reading!

SAVE this scrambled version away and use it as your main copy (although it cannot, of course, be RUN in its scrambled state).

When you want to use the program, simply load it in and type PROCLOCK. Then type in the same password as before and your program will be 'un-scrambled'. If you type in the wrong password, the program will be scrambled again and at this stage the program cannot be recovered, so take care not to forget the password.

Note that &D0 at line 32090 represents the length (plus one) of the LOCK procedure and is included in order to prevent the LOCK procedure itself being scrambled! Do be careful to type the procedure in EXACTLY as listed, otherwise PROClock itself may become scrambled. This will produce error messages, and program recovery will not generally be possible.

You can try out the LOCK procedure by typing in the whole program which comprises the lock procedure together with three additional lines representing a user program. Then proceed as follows (press Return after each line):

```
LIST
RUN
```

The three line program is listed, together with the lock procedure, and runs correctly.

```
PROClock
LIST
```

The first three lines are now scrambled, and the program, if saved in this form, is effectively locked - i.e. no one else can use it.

```
PROClock
LIST
RUN
```

Once again, after unlocking, the program lists and runs correctly.

SCREEN DISPLAY COMPETITION RESULTS

In the April isue of ELBUG (Vol.1, No.5, p.11) we set a competition in the "Saving and Loading of Screen Displays" article. What we wanted to see was what other Electron users could do with their computer display facilities. As promised the best results of the competition, which closed on April 30th, are published in this issue of the magazine.

Standards were very high but after some deliberation, the winner chosen was:

Kevin Allen

Congratulations to Kevin who will be receiving a SIR ROM expansion board for his Electron in due course as prize for his considerable efforts. We would also like to commend the effort of Mr.F.Lendon whose display is also shown here. It's a great shame that only the most extravagant of print qualities might do justice to the winning picture and unfortunatley we cannot offer that here.

Keep your eyes open for other ELBUG competitions.

Kevin Allen's winning entry.

Vauxhall Astra car by F.Lendon.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

PHASE SELECTION IN MICROPOWER'S "CROAKER" - J.Brunyee & J.Spinks

If you find Program Power's Croaker too easy to start with, then during the part of the program that displays which phase of the game you are starting, press Escape as many times as the phase that you wish to play.

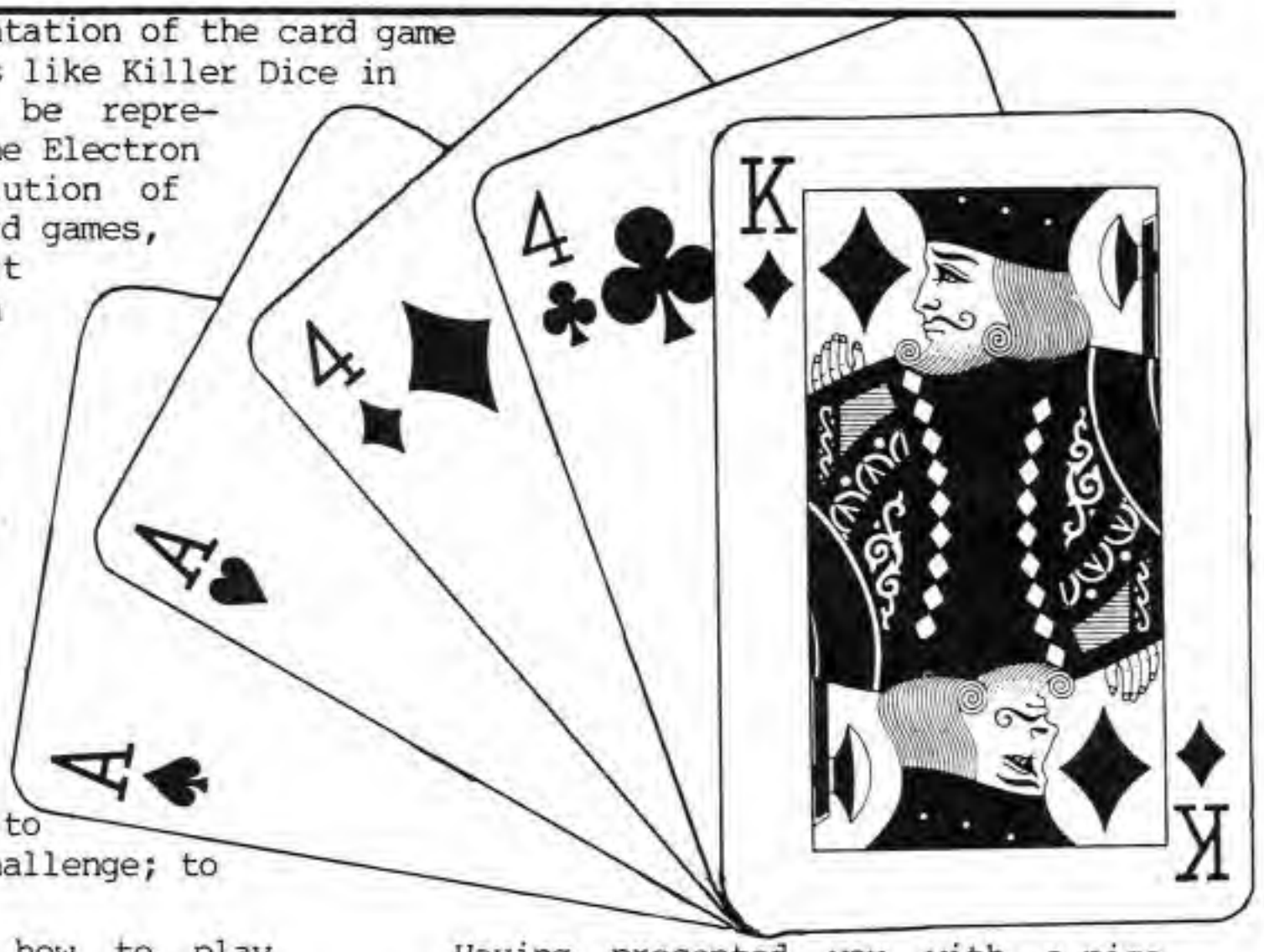
A ROUNDING ERROR? - G.Shally

This should return 1 but it actually returns 0:

```
PRINT (40.21*100) MOD 10
```

PONTOON

by Nick Day



Here is a fun implementation of the card game Pontoon or 21. Card games like Killer Dice in Elbug Vol.1 No.4, can be represented very well on the Electron with its high resolution of graphics. If you like card games, you will find the effort of typing this program in will be well rewarded.

Having presented you with a nice baize cloth, the computer is ready to preside as banker and adjudicator over a game of Pontoon with you. All the card shuffling, dealing and betting is handled by the program offering no opportunity for you to cheat! So this is your challenge; to beat the Electron.

If you do not know how to play Pontoon, the idea of the game is to win as much money as possible from the 'Banker'. Your hand (which may consist of between two and five cards) must total as near to 21 with their face values as possible without actually exceeding 21; an ace counts as 1 or 11 (the choice is yours). The rules of this version of Pontoon are that the best possible hand is a 'Pontoon' which is an ace and a picture card (11 + 10) which will give you 21. The second highest hand is a five card trick, and is accomplished by getting five cards whose total is not more than than 21, for example ace-4-6-3-4 would give you 18. The next best hands are 21 with an ace and 10, then 21 with more than 2 cards. After that the next best hand is 20, then 19 etc. The number of cards is unimportant other than in a Pontoon itself, and clearly, in five card tricks.

You are asked to place your bet when you are given your first card. Ypu should determine the amount that you bet by examining your first card, an ace would be a larger bet than, say, a 5. After betting on the first card (just press a key between 1 and 9 to

Having presented you with a nice baize cloth, the computer is ready to preside as banker and adjudicator over a game of Pontoon with you. All the card shuffling, dealing and betting is handled by the program offering no opportunity for you to cheat! So this is your challenge; to beat the Electron.

you want another card you can 'buy' this by betting any amount from 1 up to the amount of your previous bet on this hand, or you could simply 'twist' the next card which means you can have the card, but you may not increase your bet any further.

If you get over 21 you 'bust' and the banker wins regardless. When you have finished your turn, and have 'stuck' with your cards, the computer will take its turn to play. The computer as banker need only achieve a total equal to or greater than your hand to win. If the computer 'sticks' on a lower total than you, or 'busts', then you win.

PROGRAM NOTES

To save space the procedure names

have been shortened, but the most important procedures are outlined below.

PROCse Sets up the sound envelopes, variables and characters.
PROCshu Shuffles the 52 cards.
PROCsh Decides which card to display.
PROC2 to PROCk draws the corresponding card, for example:
PROCj Draws the Jack.
PROCq Draws the Queen.
PROCk Draws the King.

Take care when typing in this program, especially with the three procedures PROCj, PROCq and PROCk, otherwise you will be confronted with very strange cards indeed! For further reference on the method used to draw these cards see the article on Electron Graphics in Elbug Vol.1 No.5.

```
  10 REM PROGRAM PONTOON
  20 REM VERSION E0.2
  30 REM AUTHOR  N.DAY
  40 REM ELBUG   JULY 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 ON ERROR GOTO 420
 110 MODE1
 120 PROCse
 130 REPEAT
 140 CLG
 150 IFH%<1 VDU4:PRINTTAB(1,30);"you h
ave run out of money!":END
 160 GCOL0,0
 170 P%=0:a=0:b=0:t=0:I%=0:w=0:p=0:F%=
0:bpon=0:bust=0
 180 IFP(1)MOD13=P(3)MOD13 temp=P(3):P
(3)=P(4):P(4)=temp
 190 FORC%=1TO10 :B(C%)=(P(C%)MOD13)+1
:S(C%)=(P(C%)DIV13)+1:IFS(C%)=5 S(C%)=4
 200 NEXT
 210 FORC%=1TO12:N(C%)=0:NEXT
 220 FORD%=1TO12
 230 *FX15,1
 240 PROCki
 250 IFD%<8 ORD%>9 P%=P%+1:L(P%)=P(1)
 260 PROCp
 270 PROCf
 280 N(D%)=N%
 290 IFD%<8:IFD%=1 ORD%=3 ORD%>4:IFN%=
1 a=TRUE
 300 PROCsh
 310 IFN(D%)>10 N(D%)=10
 320 IFD%=2 PROCbet
 330 IFD%>3 ANDD%<8 PROCscore
 340 IFD%>8 ANDNOTbust PROCbank
 350 NEXT:PROCr:PROCi:PROCrep
 360 VDU4:PRINTTAB(1,30);"You have "H%
;" pound";:IFH%>1PRINT;"s":VDU5
 370 PROCm("ANOTHER HAND?")
 380 REPEAT:A%=GET:UNTILA%=89 ORA%=78:
UNTILA%=78
 390 VDU4:PROCm(" "):VDU4:PRINTTAB(1,3
0);"You finished with "H%;" pound";:IFH
%>1 PRINT;"s"
 400 END
 410 :
 420 ON ERROR OFF:MODE 6
 430 *FX4
 440 *FX12
 450 IF ERR=17 END
 460 REPORT:PRINT" at line ";ERL
 470 END
 480 :
1000 DEF PROCse
1010 p=0:bpon=0
1020 ENVELOPE1,1,0,100,0,50,2,2,5,100,
0,126,90,100
1030 ENVELOPE2,2,0,40,5,-10,5,5,10,10,
0,-100,100,40
1040 ENVELOPE3,3,0,0,0,1,1,1,50,-4,0,-
4,126,80
1050 *FX11,0
1060 *FX4,1
1070 *FX202,32
1080 H%=50
1090 DIMB(10),S(10),L(10),N(12)
1100 VDU23;8202;0;0;0;
1110 VDU24,48;128;1232;960;
1120 VDU19,2,2;0;
1130 GCOL0,130
1140 CLG
1150 VDU23,224,24,60,126,126,255,255,2
19,60
1160 VDU23,225,24,60,90,255,255,90,60,
126
1170 VDU23,226,24,60,126,255,255,126,6
0,24
```

```
 1180 VDU23,227,102,255,255,255,255,126
,60,24
 1190 VDU23,228,76,82,82,82,82,82,82,76
 1200 VDU23,229,255,255,255,255,255,255
,255,255
 1210 G%=0
 1220 PROCshu
 1230 G%=G%+1
 1240 ENDPROC
 1250 :
 1260 DEF PROCsh
 1270 SOUND0,1,4,4
 1280 GCOL0,3:MOVEX%,Y%:MOVEX%+160,Y%:P
LOT85,X%,Y%+256:PLOT85,X%+160,Y%+256:VD
U5
 1290 IFD%=2 ORD%=4 PROCb:ENDPROC
 1300 IFS%<3 GCOL0,1 ELSE GCOL0,0
 1310 MOVEX%,Y%+250:PRINTN$:IFN%<11 MOV
EX%+128,Y%+32:PRINTN$
 1320 IFN%=1 PROC5
 1330 IFN%=2 ORN%=3 PROC2
 1340 IFN%>3 ANDN%<11 PROC4
 1350 IFN%=11 PROCj
 1360 IFN%=12 PROCq
 1370 IFN%=13 PROCk
 1380 ENDPROC
 1390 :
 1400 DEFPROCf
 1410 IFD%=8 N%=N(2):S%=S(2):GOTO1460
 1420 IFD%=9 N%=N(4):S%=S(4):GOTO1460
 1430 N%=(P(1)MOD13)+1
 1440 S%=(P(1)DIV13)+1:IFS%=5 S%=4
 1450 FORC%=1TO51:P(C%)=P(C%+1):NEXT
 1460 RESTORE1510
 1470 FOR PIP=1TO13
 1480 READ NUMBER$
 1490 IFN%=PIP N$=NUMBER$
 1500 NEXT
 1510 DATAA,2,3,4,5,6,7,8,9,10,J,Q,K
 1520 IFN$="10" N$=CHR$(228)
 1530 IFS%=1 S$=CHR$(227)
 1540 IFS%=2 S$=CHR$(226)
 1550 IFS%=3 S$=CHR$(225)
 1560 IFS%=4 S$=CHR$(224)
 1570 ENDPROC
 1580 :
 1590 DEF PROC2:MOVEX%+64,Y%+192:PRINTS
$:MOVEX%+64,Y%+96:PRINTS$:IFN%=3  MOVEX
%+64,Y%+146:PRINTS$
 1600 ENDPROC
 1610 :
 1620 DEFPROC4:MOVEX%+32,Y%+224:PRINTS$
:MOVEX%+32,Y%+64:PRINTS$:MOVEX%+96,Y%+6
4:PRINTS$:MOVEX%+96,Y%+224:PRINTS$
 1630 IFN$=CHR$(228) PROC9:PROC2:ENDPROC
 1640 IFN$="5" OR N$="9" PROC5
 1650 IFN$="6"OR N$="7" ORN$="8" PROC6
 1660 ENDPROC
 1670 :
 1680 DEF PROC5:MOVEX%+64,Y%+146:PRINTS$
 1690 IFN$="9" PROC9
 1700 ENDPROC
 1710 :
 1720 DEFPROC9:MOVEX%+32,Y%+175:PRINTS$
:MOVEX%+32,Y%+116:PRINTS$:MOVEX%+96,Y%+
116:PRINTS$:MOVEX%+96,Y%+175:PRINTS$:EN
DPROC
 1730 :
 1740 DEFPROC6:MOVEX%+32,Y%+146:PRINTS$
:MOVEX%+96,Y%+146:PRINTS$:IFN$="7" ORN$
="8" PROC7
 1750 ENDPROC
 1760 :
 1770 DEFPROC7
 1780 MOVEX%+64,Y%+110:PRINTS$
 1790 IFN$="8" MOVEX%+64,Y%+187:PRINTS$
 1800 ENDPROC
 1810 :
 1820 DEFPROCj:GCOL0,0:MOVEX%+32,Y%+32:
MOVEX%+32,Y%+112:PLOT85,X%+88,Y%+32:MOV
EX%+112,Y%+32:MOVEX%+112,Y%+96:PLOT85,X
%+96,Y%+60:PLOT85,X%+96,Y%+128
 1830 GCOL0,1:MOVEX%+92,Y%+128:MOVEX%+9
2,Y%+80:PLOT85,X%+64,Y%+128:MOVEX%+32,Y
%+128:MOVEX%+40,Y%+128:PLOT85,X%+96,Y%+
32:PLOT85,X%+112,Y%+32:MOVEX%+40,Y%+32:
MOVEX%+80,Y%+32
 1840 PLOT85,X%+40,Y%+80:MOVEX%+120,Y%+
176:MOVEX%+120,Y%+32:PLOT85,X%+128,Y%+1
76:PLOT85,X%+128,Y%+32
 1850 GCOL0,0:MOVEX%+64,Y%+128:MOVEX%+4
0,Y%+128:PLOT85,X%+64,Y%+192:PLOT85,X%+
48,Y%+192
 1860 GCOL0,1:MOVEX%+112,Y%+192:PLOT85,
X%+128,Y%+224:MOVEX%+48,Y%+192:PLOT85,X
%+32,Y%+224
 1870 GCOL0,0:MOVEX%+76,Y%+144:DRAWX%+9
2,Y%+144:MOVEX%+80,Y%+156:DRAWX%+88,Y%+
156:DRAWX%+88,Y%+168:MOVEX%+80,Y%+184:D
RAWX%+72,Y%+176:MOVEX%+102,Y%+184:DRAWX
%+96,Y%+176
 1880 GCOL0,1:MOVEX%+104,Y%+202:MOVEX%+
112,Y%+224:PLOT85,X%+56,Y%+202:PLOT85,X
%+48,Y%+224:PROCPIC:ENDPROC
 1890 :
 1900 DEF PROCPIC
 1910 GCOL0,0
 1920 MOVEX%+32,Y%+32:DRAWX%+128,Y%+32:
DRAWX%+128,Y%+224:DRAWX%+32,Y%+224:DRAW
X%+32,Y%+32
 1930 IFS%<3 GCOL0,1
 1940 MOVEX%+132,Y%+64:PRINTN$:MOVEX%,Y
%+218:PRINTS$:MOVEX%+135,Y%+32:PRINTS$
 1950 ENDPROC
 1960 :
 1970 DEFPROCk:GCOL0,1:MOVEX%+32,Y%+224
:MOVEX%+48,Y%+192:PLOT85,X%+128,Y%+224:
PLOT85,X%+112,Y%+192
```

→

```
 1980 GCOL0,0:FORH=112TO80STEP-8:MOVEX%
+H,Y%+192:DRAWX%+H+16,Y%+128:NEXT:FORH=
88TO64STEP-8:MOVEX%+H,Y%+144:DRAWX%+H+1
6,Y%+128:NEXT
 1990 MOVEX%+80,Y%+152:DRAWX%+60,Y%+152
:MOVEX%+72,Y%+160:DRAWX%+60,Y%+152:MOVE
X%+72,Y%+160:DRAWX%+68,Y%+180:DRAWX%+52
,Y%+180:MOVEX%+80,Y%+180:DRAWX%+64,Y%+1
80:MOVEX%+48,Y%+192:DRAWX%+72,Y%+128:MO
VEX%+32,Y%+112:PLOT85,X%+32,Y%+32
 2000 MOVEX%+72,Y%+128:PLOT85,X%+128,Y%
+128
 2010 PLOT85,X%+128,Y%+32:PLOT85,X%+32,
Y%+32:GCOL0,1:MOVEX%+32,Y%+64:MOVEX%+12
8,Y%+128:PLOT85,X%+32,Y%+32:PLOT85,X%+1
28,Y%+96:GCOL0,3
 2020 MOVEX%+44,Y%+220:PRINT"X":MOVEX%+
68,Y%+220:PRINT"X":MOVEX%+92,Y%+220:PRI
NT"X"
 2030 MOVEX%+48,Y%+32:MOVEX%+64,Y%+32:P
LOT85,X%+48,Y%+128:PLOT85,X%+64,Y%+128:
MOVEX%+112,Y%+128:MOVEX%+112,Y%+32:PLOT
85,X%+96,Y%+128:PLOT85,X%+96,Y%+32:GCOL
0,0:FORH=132TO64STEP-16:MOVEX%+44,Y%+H:
PRINT",":MOVEX%+92,Y%+H:PRINT",":NEXT
 2040 PROCPIC:ENDPROC
 2050 :
 2060 DEFPROCq:GCOL0,0:MOVEX%+32,Y%+224
:MOVEX%+32,Y%+128:PLOT85,X%+96,Y%+224:M
OVEX%+112,Y%+224:PLOT85,X%+112,Y%+176:M
OVEX%+84,Y%+192:DRAWX%+72,Y%+168:MOVEX%
+104,Y%+168:DRAWX%+92,Y%+192:DRAWX%+96,
Y%+148:DRAWX%+84,Y%+148
 2070 GCOL0,1:MOVEX%+92,Y%+140:DRAWX%+7
6,Y%+140:MOVEX%+56,Y%+224:MOVEX%+72,Y%+
224:PLOT85,X%+32,Y%+160:PLOT85,X%+32,Y%
+128:MOVEX%+48,Y%+128:PLOT85,X%+32,Y%+3
2:PLOT85,X%+48,Y%+32
 2080 FORH=48TO64STEP8:MOVEX%+H,Y%+32:D
RAWX%+H+32,Y%+96:NEXT:MOVEX%+96,Y%+128:
MOVEX%+64,Y%+128:PLOT85,X%+128,Y%+112:P
LOT85,X%+128,Y%+32
 2090 GCOL0,0:MOVEX%+48,Y%+112:PRINT"*"
:MOVEX%+80,Y%+64:PRINT"*":MOVEX%+96,Y%+
128:MOVEX%+112,Y%+64:PLOT85,X%+128,Y%+1
12:PROCPIC:ENDPROC
 2100 :
 2110 DEFPROCb:GCOL0,1:FORC%=40TO244STE
P20:FORW%=12TO128STEP20:MOVEX%+W%,Y%+C%
:PRINT"#":NEXT,:ENDPROC
 2120 :
 2130 DEFPROCshu:VDU4:PRINTTAB(1,30);"I
'm shuffling      ":VDU5:IFG%>0 GOTO2170
 2140 DIMP(52)
 2150 FORT%=1TO52
 2160 P(T%)=T%:NEXT
 2170 FORC%=1 TO 52
 2180 dummy=P(C%)
 2190 newplace=RND(52)
 2200 P(C%)=P(newplace)
 2210 P(newplace)=dummy
 2220 NEXT
 2230 ENDPROC
 2240 :
 2250 DEFPROCp
 2260 IFD%=1X%=160:Y%=192
 2270 IFD%=2Y%=640:X%=160
 2280 IFD%=3X%=348:Y%=192
 2290 IFD%=4Y%=640:X%=348
 2300 IFD%=5X%=576
 2310 IFD%=6X%=784
 2320 IFD%=7X%=992
 2330 IFD%=8X%=160
 2340 IFD%=9X%=348
 2350 IFD%=10X%=576
 2360 IFD%=11X%=784
 2370 IFD%=12X%=992
 2380 IFD%>4 ANDD%<8 Y%=192
 2390 IFD%>7 Y%=640
 2400 ENDPROC
 2410 :
 2420 DEFPROCscore
 2430 Q%=N(1)+N(3)+N(5)+N(6)+N(7):IFa=T
RUE ANDQ%<12 Q%=Q%+10
 2440 IFQ%>21PROCm("BUST!"):D%=12:Q%=0:
bust=TRUE:ENDPROC
 2450 IFD%=4 ANDN(1)=1 ANDB(3)>10 p=TRUE
 2460 IFD%=4 ANDN(3)=1 ANDB(1)>10 p=TRUE
 2470 IFN(7)>0 PROCm("5 CARD TRICK"):EN
DPROC
 2480 IF w=TRUE AND Q%>15 PROCm("STICK
or TWIST?")ELSE IF w=TRUE PROCm("TWIST?
")ELSE IF Q%>15 PROCm("STICK,TWIST,BET?
") ELSE PROCm("TWIST or BET?")
 2490 REPEAT
 2500 A%=GET
 2510 UNTILA%=83 ORA%=84 ORA%=89 ORA%=66
 2520 IF w=TRUE ANDA%=66 GOTO2490
 2530 IFA%=83 AND Q%>15 D%=7:PROCm("YOU
 STICK"):ENDPROC ELSE IF A%=83 VDU 7:GO
TO2430
 2540 IFA%=84 OR A%=89 w=TRUE:PROCm("TW
IST"):ENDPROC
 2550 IFA%=66 PROCbet:ENDPROC
 2560 :
 2570 DEFPROCbank
 2580 IFN(2)=1 ORN(4)=1 b=TRUE
 2590 IFD%>7 ANDN%=1 b=TRUE
 2600 R%=N(2)+N(4)+N(10)+N(11)+N(12):IF
b=TRUE ANDR%<12 R%=R%+10
 2610 IFD%=9 ANDN(2)=1 ANDB(4)>10 Q%=0:
bpon=TRUE:D%=12:ENDPROC
 2620 IFD%=9ANDN(4)=1 ANDB(2)>10 Q%=0:b
pon=TRUE:D%=12:ENDPROC
 2630 IFp ANDNOTbpon PROCm("PAY PONTOON
!"):D%=12:ENDPROC
 2640 IFR%>21PROCm("DEALER'S BUST"):D%=
12:R%=0:ENDPROC
 2650 IFR%<17 ANDD%=11 ANDN(7)>0 ENDPROC
```

→

```
 2660 IFR%<17 ANDRND(1)>.5 AND w=TRUE:I
FN(6)>0 ORN(5)>0 ENDPROC
 2670 IFR%=21 ANDNOTp AND N(7)=0 D%=12:
ENDPROC
 2680 IFR%>15 PROCm("BANK STICKS!"):VDU
4:PRINT;"PAY";R%+1:VDU5:D%=12:ENDPROC
 2690 ENDPROC
 2700 :
 2710 DEFPROCr:PROCw
 2720 IF Q%>21 OR Q%=0 I%=0:PROC1:ENDPR
OC
 2730 IFN(12)>0 AND NOT p I%=0:PROC1:EN
DPROC
 2740 IFN(7)>0 ANDNOTbpon  H%=H%+(I%*3)
:PROCv:ENDPROC
 2750 IFR%=21 ANDNOTp ANDN(7)=0 PROCm("
SORRY!"):I%=0:PROC1:ENDPROC
 2760 IFp H%=H%+(I%*4):PROCv:ENDPROC
 2770 IFQ%>R% H%=H%+(I%*2):PROCv:ENDPROC
 2780 IFQ%<R%+1 PROC1:ENDPROC
 2790 :
 2800 DEFPROCrep
 2810 FORC%=1TOP%
 2820 P(53-C%)=L(C%)
 2830 NEXT:ENDPROC
 2840 :
 2850 DEFPROCbet
 2860 PROCm("YOUR BET?")
 2870 B%=GET:IFB%<49ORB%>57GOTO2870
 2880 IFD%>3 AND B%>F%GOTO2860
 2890 IFB%-48>H% PROCm("SILLY!"):PROCw:
GOTO2850
 2900 F%=B%
 2910 B%=B%-48:I%=I%+B%:PROCi:PROCi2
 2920 H%=H%-B%:PROCm(STRING$(15,CHR$32))
 2930 ENDPROC
 2940 :
 2950 DEFPROCm(M$):PROCw:VDU4:PRINTTAB(
22,30);SPC(17):PRINTTAB(22,30);M$;:VDU5
:ENDPROC
 2960 :
 2970 DEFPROCw:FORC%=1TO5000:NEXT:ENDPR
OC
 2980 :
 2990 DEFPROCki
 3000 VDU4:PRINTTAB(1,30);"You have "H%;
" pound";:IFH%>1PRINT;"s ":VDU5:ENDPROC
 3010 :
 3020 DEFPROCi:VDU5:GCOL0,2:FORC%=480TO
800 STEP32:MOVEC%,512:PRINTCHR$(229):NE
XT
 3030 ENDPROC
 3040 :
 3050 DEFPROCi2:GCOL0,0:MOVE480,512:VDU
5:PRINT;I%;" Pound";:IFI%>1 PRINT;"s":V
DU4
 3060 ENDPROC
 3070 :
 3080 DEFPROCv:SOUND1,3,197,4:ENDPROC
 3090 :
 3100 DEFPROC1:SOUND1,3,33,4:ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

THE HANGING 'ELSE' PROBLEM - A.Williams

Nesting of IF THEN ELSE statements can give rise to ambiguities and therefore, unexpected results. A contrived example follows in this piece of program (it's spaced out a bit for clarity):

```
10 IF TRUE=FALSE THEN IF TRUE=TRUE THEN PRINT"1"
                  ELSE PRINT"2"
        ELSE PRINT"3"
```

You might expect the result to be "3" but the Electron prints "2". This is because the Basic interpreter in the Electron searches for the next occurrence of an 'ELSE' after the evaluation of 'TRUE=FALSE' (which of course is 'FALSE'). The first one that it comes across is the wrong one in effect.

PROGRAMMING CAPS LOCK MODE - R.Lambley

To ensure that the CAPS LOCK mode is selected before an answer is required from the keyboard add the *FX202,32 command to a line before inputting anything. In this way you would only have to check the input against capitals to verify that it was correct. For example:

```
100 *FX202,32
110 REPEAT INPUTTAB(5,10)"Do you want instructions (Y/N)?"reply$
120 UNTIL reply$<>""
130 IF reply$="Y" THEN PROCinstructions
```

Line 130 does not have to check for a lower case 'y' as well an upper case 'Y'.

Subscriptions and Software Help Line
St Albans (0727) 60263
Manned Mon-Fri 1pm-4pm.

BACK ISSUES AND SUBSCRIPTIONS

BACK ISSUES (Members only)

All back issues will be kept in print (from November 1983). Send 90p per issue PLUS an A5 SAE to the subscriptions address. Back copies of BEEBUG are available to ELBUG members at this same price. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the advertising supplements are not supplied with back issues.

Subscription and Software Address

ELBUG
PO BOX 109
High Wycombe
Bucks

SUBSCRIPTIONS

Send all applications for membership, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:

U.K.
£5.90 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
Eire and Europe
Membership £16 for one year.
Middle East £19
Americas and Africa £21
Elsewhere £23
Payments in Sterling preferred.

SOFTWARE (Members only)

This is available from the software address.

MAGAZINE CONTRIBUTIONS AND TECHNICAL QUERIES

Please send all contributions and technical queries to the editorial address opposite. All contributions published in the magazine will be paid for at the rate of £25 per page.

We will also pay £10 for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address

ELBUG
PO Box 50
St Albans
Herts

ELBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Production Editor: Phyllida Vanstone.
Technical Assistants David Fell, Nigel Harris and Alan Webster.
Managing Editor: Lee Calcraft.
Thanks are due to Sheridan Williams, and Adrian Calcraft for assistance with this issue.

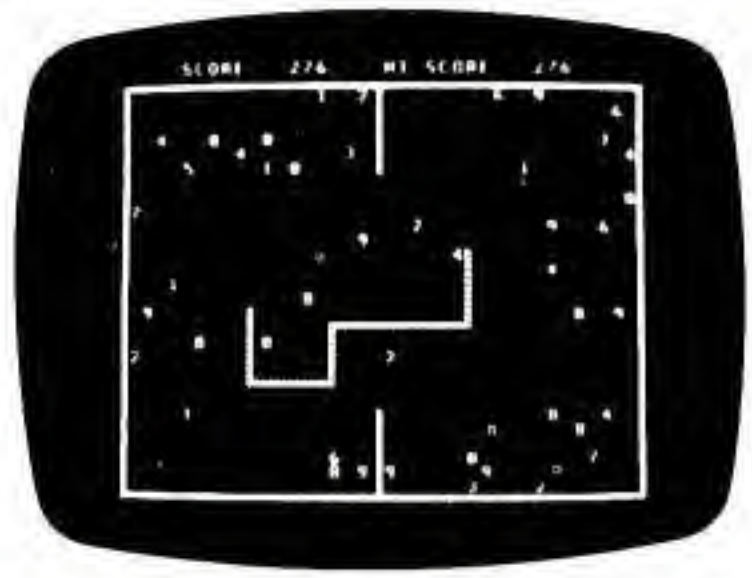
BEEBUGSOFT

Snake

Snake is a very addictive, fast moving arcade game. The purpose of the game is to direct the snake around the screen, eating up the randomly placed objects. With every bite that it takes, the tail grows longer and the snake moves faster. Just to complicate things, there are also certain scattered obstacles wnich must be avoided.

To even complete the first screen requires skill and concentration. Further frames use different shaped playing areas, which as well as adding variety, require much greater skill to complete.

This game is much more exciting than many games of a similar nature, and once played, proves to be surprisingly compulsive.



Snake costs £4.00 inc VAT. Please add 50p post & packing.
Overseas orders: £5.50 inc post & packing - VAT not charged.
Send order with membership number to:
BEEBUGSOFT, P.O. Box 109, High Wycombe, Bucks

THE BEST OF ELBUG ON CASSETTE

Many of the best programs published in ELBUG have been collected together and published by Penguin Books under the name "Games and other programs for the Acorn Electron" at £3.95. This book is part of the Penguin Acorn Computer Library and at present there is just one other title available though others are planned.

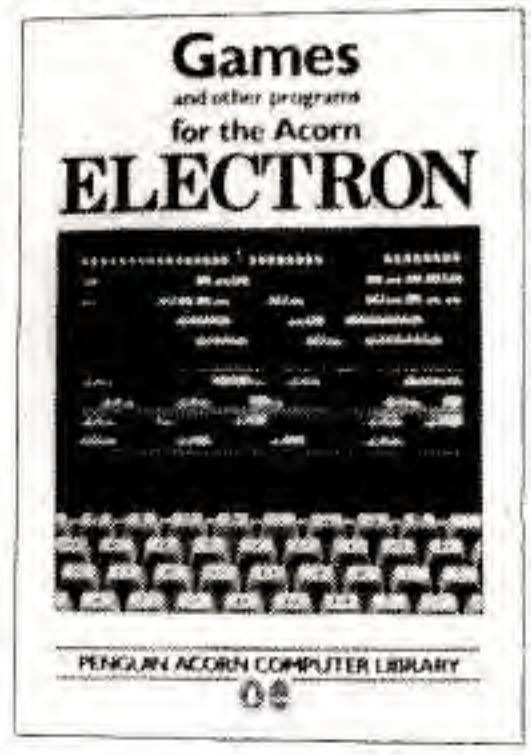
There are 20 programs in all in four different categories:

Action Games
- Munch-Man, Mars Lander, Invasion
- Robot Attack, Hedgehog

Thought games
- Higher/Lower, Five-Dice, Life
- Anagrams, Return of the Diamond

Visual Displays
- Union Jack, Square Dance, Ellipto
- Screenplay, 3-D Rotation

Utilities
- Sound Wizard, Bad Program Lister
- 3-D Lettering, Bad Program Rescue
- Double Height Text



All 20 programs are now available on cassette from our software address (in High Wycombe) price £7 to members and £9 to non-members, plus 50p post & packing in either case.

ELBUG MAGAZINE CASSETTE

To save wear and tear on fingers and brain, we offer, each month, a cassette of the programs featured in the latest edition of ELBUG. The first program on each tape is a menu program, detailing the tape's contents, and allowing the selection of individual programs. The tapes are produced to a high technical standard by the process used for the BEEBUGSOFT range of titles.

Magazine cassettes have been produced for each issue of ELBUG from Volume 1 Number 1 onwards and are all available from stock, priced £3.00 each inclusive of VAT. See below for ordering information.

ELBUG MAGAZINE CASSETTE JULY 1984

This month's cassette (Vol.1 no.8) includes:
The Flowers of Hell game, Disco Lights Display, automatic Cassette Indexer, the classic Breakout game for the Electron, Graphics Example (on manipulating colours), a useful Program Protection routine, and a colourful version of the Pontoon card game.

MAGAZINE CASSETTE SUBSCRIPTION

We are also able to offer ELBUG members subscription to the magazine cassette, this gives the added advantage of receiving the cassette at around the same time as the magazine each month. Subscriptions may either be for a period of 1 year or 6 months, however for an introductory period we are also offering a trial 3 months subscription. (NOTE Magazine cassettes are produced 10 times each year).

If required, subscriptions may be backdated as far as Volume 1 Number 1, so when applying please write to the address below quoting your membership number and the issue from which you would like your subscription to start.

MAGAZINE CASSETTE ORDERING INFORMATION

Individual ELBUG Magazine Cassettes £ 3.00
P & P: Please add 50p for the first and 30p for each subsequent cassette.
Overseas orders: Calculate the UK price including post, then deduct 15% VAT and add £1 per item.

Magazine Cassette Subscription

1 YEAR (10 issues)	£33.00 Incl.....	O'SEAS	£39.00	No VAT payable
6 MONTHS(5 issues)	£17.00 Incl.....	O'SEAS	£20.00	No VAT payable
3 MONTHS(3 issues)	£10.00 Incl.....	O'SEAS	£13.00	No VAT payable

Please be sure to specify that you require subscription to the ELBUG magazine cassette, and enclose your membersip number with a cheque made payable to BEEBUGSOFT.
Send to..
ELBUG Magazine Cassette, BEEBUGSOFT, PO Box 109, High Wycombe, HP10 8HQ.

Printed in England by Staples Printers St Albans Limited at The Priory Press.